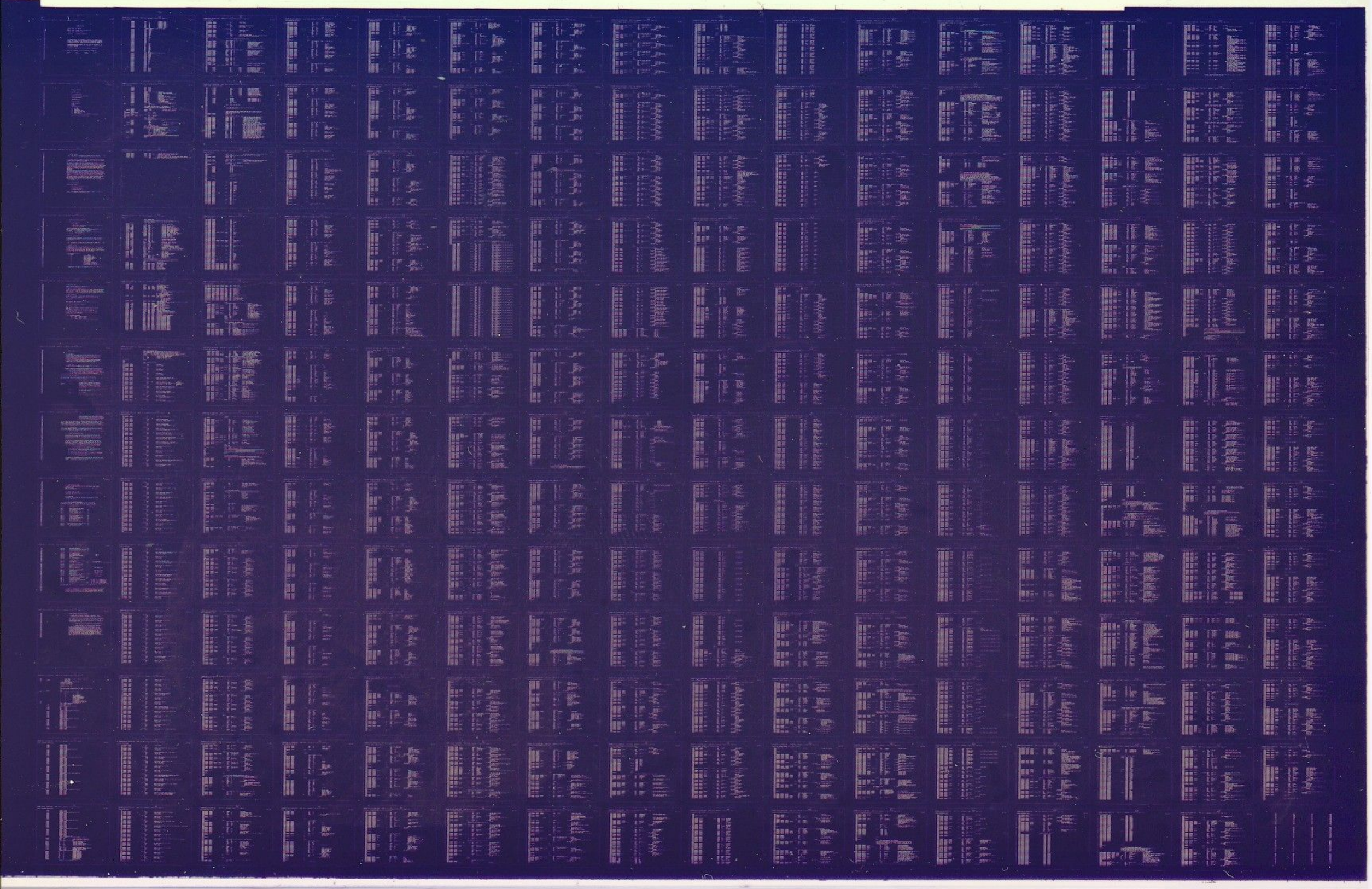
KDJ11-B

KDJ11-B CLUSTER DIAG
COKDAD0

AH-T854D-MC
1 OF 3 JUL 1985
COPYRIGHT© 1984-85 MADE IN USA

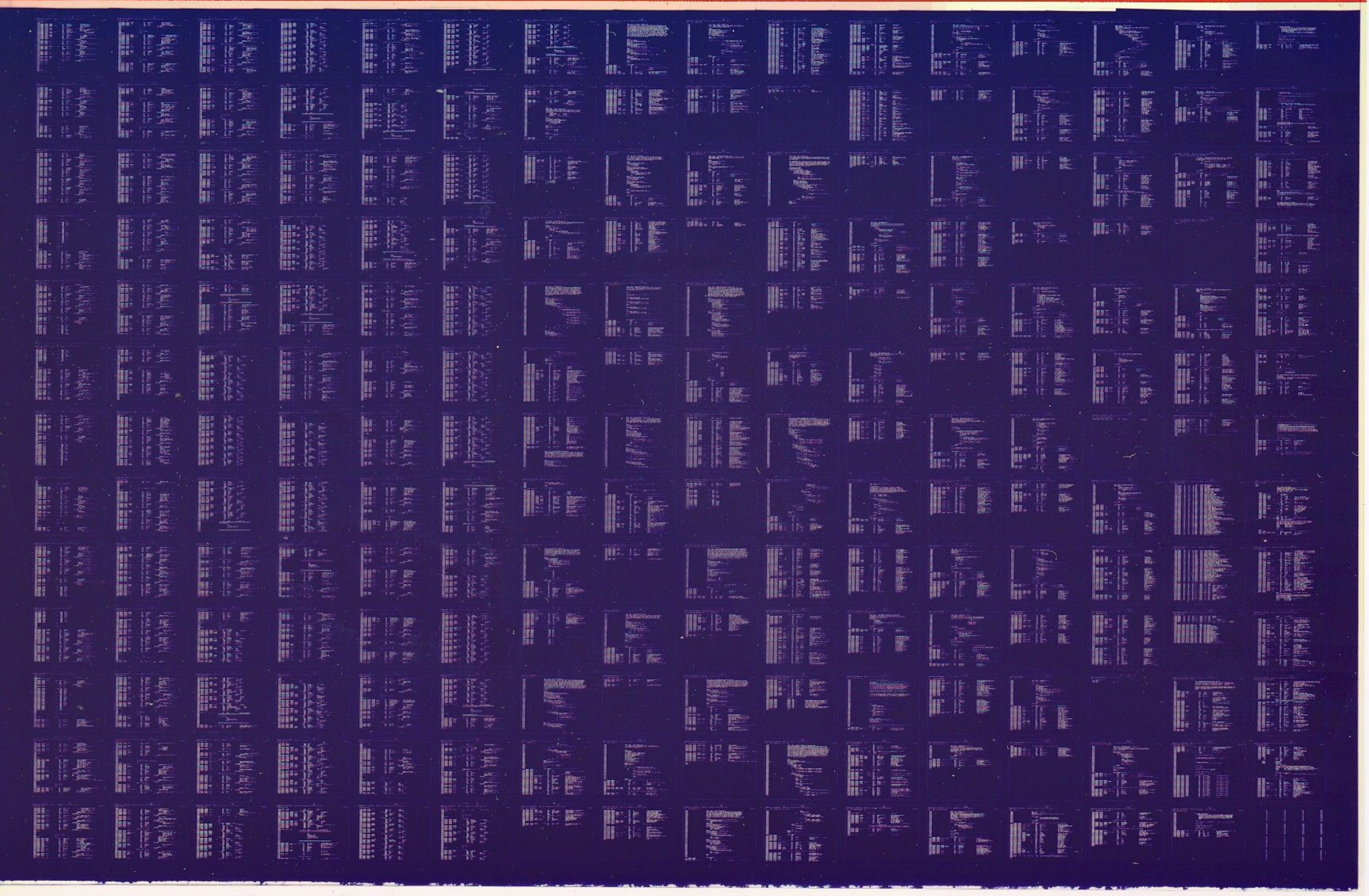


KDJ11-B

KDJ11-B CLUSTER DIAG
COKDAD0

AH-T854D-MC
2 OF 3 JUL 1985
COPYRIGHT© 1984-85

digital
MADE IN USA



KDJ11-B

KDJ11-B CLUSTER DIAG
COKDAD0

AH-T854D-MC
3 OF 3 JUL 1985
COPYRIGHT© 1984-85

digital
MADE IN USA

```
# .  W
A ;;                                                                    SEQ 000
1
COKDADO KDJ11-B CLUSTER DIAG.   MACRO Y05.02  Wednesday 27-Feb-85 17:55  Page 2

      1
      2
      3                                                          .REM &
      4
      5                                          IDENTIFICATION
      6                                          --------------
      7
      8                                  PRODUCT CODE:  AC-T853D-MC
      9
     10                                  PRODUCT NAME:  COKDADO KDJ11-B CLUSTER DIAG.
     11
     12                                  PRODUCT DATE:  FEBRUARY, 1985
     13
     14                                  MAINTAINER:    DIAGNOSTIC ENGINEERING
     15
     16
     17
     18
     19
     20                                  THE INFORMATION IN THIS DOCUMENT IS  SUBJECT  TO  CHANGE WITHOUT
     21                                  NOTICE  AND  SHOULD  NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL
     22                                  EQUIPMENT CORPORATION.  DIGITAL EQUIPMENT CORPORATION ASSUMES NO
     23                                  RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.
     24
     25                                  NO RESPONSIBILITY IS ASSUMED  FOR  THE  USE  OR  RELIABILITY  OF
     26                                  SOFTWARE ON EQUIPMENT THAT IS NOT  SUPPLIED  BY  DIGITAL  OR ITS
     27                                  AFFILIATED COMPANIES.
     28
     29                                  COPYRIGHT (C) 1984,1985 BY DIGITAL EQUIPMENT CORPORATION
     30
     31
     32                            THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:
     33
     34                                  DIGITAL          PDP              UNIBUS          MASSBUS
     35                                  DEC              DECUS            DECTAPE
     36
     37
     38
     39
     40
     41
     42
```

TABLE OF CONTENTS

1. PROGRAM ABSRACT

2. SYSTEM REQUIREMENTS

3. LOADING AND STARTING PROCEDURES

4. SPECIAL ENVIRONMENTS

5. PROGRAM OPTIONS

6. EXECUTION TIMES

7. ERROR INFORMATION

8. EXAMPLES

9. PROGRAM DESCRIPTION

   9.1 J11 CODE
   9.2 CACHE CODE
   9.3 ON-BOARD ROM CODE
   9.4 LINE TIME CLOCK CODE
   9.5 SERIAL LINE UNIT CODE
   9.6 Q22BE CODE
   9.7 LIST of SUBTESTS showing CACHE/APT dependency

10. PROGRAM UPDATES AND MODIFICATIONS

```
 79
 80                    1.   PROGRAM ABSTRACT
 81
 82                      ****   see special instructions for CACHE testing under APT,     ****
 83                      ****   section 9.2, and EEROM testing under APT, section 9.3     ****
 84
 85
 86                    This program tests out KDJ11-B CPU board, including the J11 chip set,
 87                    on-board cache, on-board ROM's, including 16 bit and the 8 bit EEROM,
 88                    serial line unit, and line time clocks.
 89
 90                    The KDJ11-B is a PDP-11 CPU that incorporates the J11 chip set as the
 91                    heart of the processor. It is a quad height Q22 bus module. It has
 92                    on-board cache, some of the functionality of the cache is hidden inside
 93                    the J11 and the rest of the functions implemented in two on-board gate
 94                    arrays. The storage capacity of the cache is 4K bytes of RAM, called
 95                    data RAM's. The cache is implemented as a direct mapped cache with
 96                    address bits 21 through 13 stored in a different set of RAM's called
 97                    tag store.
 98                    The KDJ11-B also has two on-board ROM's. One of them, the 16-bit
 99                    addressable ROM, contains the self-test and the boot codes. The other
100                    ROM, the 8-bit addressable one, contains the base area with hardware
101                    selection parameters, optional bootstraps, optional UFD (User Friendly
102                    Diagnostic) system desctiption area, and optional foreign language
103                    text.
104                    The Serial Line Unit is implemented thru a Dlart chip which provides
105                    the standard console interface to the CPU. It has internal loop back
106                    mode and provides with three clock lines: 800HZ, 60HZ, and 50HZ.
107                    The line time clock functions are implemented using those three lines
108                    and the BEVENT line. The line clock status register is implemented in
109                    one of the gate arrays.
110
111
112                    2.  SYSTEM REQUIREMENTS
113
114                    Hardware Requirements
115
116                    To run successfully the diagnostic needs:
117
118                            1. KDJ11-B CPU module
119                            2. console terminal
120                            3. at least 28K of memory
121
122                    In DVT,  and stage one manufacturing (module assembly) the Q22 Bus
123                    exerciser is needed to check Q22 Bus logic.
```

```
125
126                                   3.  LOADING AND STARTING PROCEDURES
127
128                                   To start-up this program:
129
130                                           1. Boot XXDP+
131
132                                           2. Type "R NAME", where name is the name of the BIN or BIC
133                                              file for this program.
134
135                                   The starting address of the program is 200.
136                                   Note: if trying to restart the program in an arbitrary place after
137                                   HALT on Break the following registers should be set up:
138                                   17777572=0        to disable memmory management
139                                   17777520=1000     to clear diagnostic mode (bit 8), but still save
140                                                     HALT on Break
141                                   17777746=400      to flush the cache
142
143
144                                   4.  SPECIAL ENVIRONMENTS
145
146                                   The program is APT compatible. It can also be run under the UFD monitor.
147                                   In those cases none of the standard error printouts occur. Refer
148                                   to corresponding documents on running procedures in APT and under
149                                   UFD monitor.
150
151                                     ****   see special instructions for CACHE testing under APT,     ****
152                                     ****   section 9.2, and EEROM testing under APT, section 9.3     ****
153
154                                   5.  PROGRAM OPTIONS
155
156                                   The Q22 Bus Exerciser is utilized if it is present in the system and
157                                   the diagnostic is not running in UFD mode.
158                                   Standard capabilities of looping on test and on error are provided.
159                                   In order to run the extensive cache data RAM test bit 7 has to be set
160                                   in the software switch register. To run the extensive cache tag RAM test
161                                   bit 6 has to be set in the software switch register.
162
163
164                                   SWITCH REGISTER SELECTION:
165                                       BIT NUMBER                          USE
166                                            15                     HALT ON ERROR
167                                            14                     LOOP ON PRESENT TEST
168                                            13                     INHIBIT ERROR TYPEOUTS
169                                   ***      12                     EEROM subtest RUN switch
170                                            11                     INHIBIT ITERATIONS
171                                            10                     BELL ON ERROR
172                                             9                     LOOP ON ERROR
173                                             8                     LOOP ON TEST IN SWR<5-0>
174                                             7                     DO EXTENSIVE DATA RAM TEST
175                                             6                     DO EXTENSIVE TAG RAM TEST
176                                           5-0                     Subtest number to loop on (BIT 8)
177
178                                   *** running with this test will change all but the first 109 bytes of
179                                       EEROM data!  be SURE to read instructions in sect. 9.3
```

```
181                                     6. EXECUTION TIMES
182
183                                     Without extensive RAM tests, the diagnostic runs in under 15 seconds.
184                                     With them, it takes about 2 minutes.
185
186                                     In addition, when the EEROM test bit is on, pass 1 takes an additional
187                                     amount of time of 1 minute for 15MHZ J-11, to about 40 sec. for the
188                                     20MHZ. version.  These test times are for the 2K EEROM, for 8K parts,
189                                     multiply by 4.  If running under APT, the system manager must be sure
190                                     the first pass run time allows adequate time for completion of this test!
191
192                                     7. ERROR INFORMATION
193
194                                     In the case of errors, a failing PC and test numbers are given.
195                                     Where it is possible, expected and received data are given.
196                                     For an example, see section 8.
197
198                                     8. EXAMPLES
199
200                                     After booting XXDP+ and starting the program, the following
201                                     will appear on the terminal:
202
203                                     * KDJ11-B CPU DIAGNOSTIC - COKDADO *
204
205                                     SWR = XXXXXX    NEW =
206
207                                     where XXXXXX correspond to present software switch register
208                                     setting.
209                                     After "NEW" an operator can do one of the following:
210                                     1) type in a new software switch register setting followed
211                                     by carrige return or
212                                     2) just type in carriage return in which case the software
213                                     register will remain unchanged.
214
215                                     Example of error printout:
216                                             ERROR IN TAG STORE
217                                              TEST   ERROR   ADDRESS ADDRESS
218                                               #      PC     <21-16> <15-0>
219                                               27    105620  66600   000000
220                                     Note: this may not correspond to the actual Program Counter.
221
```

```
223
224                                   9. PROGRAM DESCRIPTION
225
226
227                                   9.1  J11 CODE
228                                   This portion of the code tests out the J11 chip set. It is broken
229                                   into 3 pieces: CPU tests, which verify different instructions in
230                                   different modes and different trap conditions; MMU tests, which
231                                   verify different functions of MMU; and FFP tests, which do
232                                   different floating point instructions.
233                                   This portion of the code have been written in close relationship
234                                   with the J11 microcode. Therefore, even though not all possible
235                                   instructions in all possible addressing mode have been tested, an
236                                   attempt has been made to exercise all of the microcode.
237
238
239                                   9.2  CACHE CODE
240                                   This portion of the diagnostic verifies all cache functions.
241                                   Data and tag RAM's tests are also included.
242
243                           ************************************************************************
244                                   Note: in order to run extensive cache RAM's tests the corresponding
245                                   bits in the software switch register should be set. See section 5.
246                                   ----In particular, the number in $USWR (sware reg #2) should equal
247                                       the "first pass run time" set up at installation---- (the exception
248                                           is when $USWR is set to ZERO, no condition applies)
249                           ************************************************************************
250
251                           TESTING CACHE FUNCTIONS UNDER APT
252
253                           The testing of Cache related functions under APT is facilitated
254                           by use of $USWR containing a number which will indicate the number
255                           of passes with inclusive cache tests as follows:
256
257                                   (all numbers in decimal, $USWR loaded by APT manager )
258
259                                                                  one pass with all subtests followed by
260                           default       $USWR = 0                continuous testing of non-cache dependent
261                           setting                                subtests until APT Break is encountered
262                                                                  (APT system must not BREAK in for this
263                                                                   "First Pass Run Time", now assumed to be
264                                                                   approx 16 sec. )
265
266
267                                         $USWR < 16               only non-cache dependent tests will be run
268                                                                   APT may break anytime
269
270                                         $USWR = 16               same as $USWR = 0
271
272                                         $USWR > 16               $USWR is divided by 16 (pass time ) to get a
```

number of passes which include all subtests (including cache) After this number of passes, continuous testing with only non-cache dependent sub-tests is continued during which time APT may break with no bad effects.

In this case, the APT setup must not attempt to break for a length of time equal to $USWR contents, in seconds. The nominal 16 seconds per pass, and the actual pass time of about 10 seconds (which changes as tests may be added) allow a safty margin adding additional tests. The actual number of passes done will be equal to $USWR divided by 16, (if not 0, or < 16).

!!!!!!!!!!!!!!

***** The guarrantee that APT will not attempt to break into the diagnostic for a given number of seconds is provided by the APT system manager loading the variable $PASTM with the same number $USWR received.
If this is not done, the message "hung diagnostic" will likely be received*******

!!!!!!!!!!!!!!

9.3 ON-BOARD ROM AND EEROM CODE

These tests verify the checksums of the 16-bit ROM and the base area of the 8-bit EEROM, and a write and read of 1's and 0's to each location of the EEROM. The EEROM may be affected by multiple writes, (>10,000 cycles) so it is advisable to select this test with care.

The testing of the on-board EEROM is accomplished by the setting of bit 12 in the software switch register, which will see that the EEROM test is run once. (Pass 1) It should be noted that this bit may have another use in the SYSMAC macro $EOP, but is not so used within this program. The BCSR is also set for Halt on Break on during this test, as a troubleshooting aid in stand-alone modes.

*** WHEN RUNNING THIS TEST UNDER APT ****

After loading and starting the program with EEROM test switch on, APT must not (break) interrupt the test until the time given in section 6, EXECUTION TIMES, has passsed.

This test should never be "scripted" in a way which would cause the test to be run multiple times. It should be run instead once at the begining or end of a script.

****

!! The contents of the EEROM are lost, except for the 109 (decimal) byte
!! representing the SETUP area, which are restored at the end of the EEROM
!! subtest. If the test is interrupted before these locations are re-
!! stored, unpredictable results may occur.

9.4 LINE TIME CLOCKS CODE

This part of the program verifies the functions of all 4 clock lines: 3 of them from the serial line chip (50HZ, 60HZ, 800HZ) and BEVENT line.
Note: in UFD mode only functions corresponding to Boot Rom selection are checked.

9.5 SERIAL LINE UNIT CODE

These tests verify the functionality of the SLU chip utilizing the maintenance mode of the chip.

9.6 Q22BE CODE

These tests verify interrupt arbitration of the KDJ11-B, DMA protocol, and cache functions related to the DMA activities, including PMG counter.

9.7 LIST OF SUBTESTS PERFORMED

The following list represents the sequential order of subtests performed in cokdacA0..... subtests which are subject to the APT qualifications of section 9.2 are indicated by a Cache-APT label.

TEST NO.	TEST NAME/FUNCTION	APT-CACHE sel	NOTES.
test 1	base instruction set tests	.....	
test 2	check (cache) register access	yes	
test 3	CCR register bit test	yes	
test 4	force miss test	yes	
test 5	hit/miss register test part 1	yes	
test 6	hit/miss register test	yes	
test 7	byte allocation test	yes	
test 10	PDR bit 15 (bypass) test	yes	
test 11	flush cache test	yes	
test 12	unconditional bypass test	yes	
test 13	write wrong data parity test	yes	
test 14	write wrong tag parity	yes	
test 15	parity abort test	yes	
test 16	parity interrupt test	yes	
test 17	miscellaneous parity test	yes	
test 20	memory system error register test	yes	
test 21	check parity aborts blocked	yes	
test 22	multi-processing instruction	yes	
test 23	data store ram, tests	yes	

```
                        test 24         tag store ram tests                 yes
                        test 25         standalone mode tests               yes
                        test 26         moving inversions test data RAM     yes
                        test 27         moving inversions for tag store     yes

                        test 30         PCR read/write bits                 .......
                        test 31         bcsr read/write bits                .......
                        test 32         16 bit ROM checksum                 .......     HOB on
                        test 33         8 bit EEROM checksum test           .......     HOB on
                        test 34         EEROM checkerboard mem test         .......     note 1.0
                        test 35         LKS bit 7                           .......
                        test 36         LKS interrupt priority              .......
                        test 37         line clock disable                  .......

                        test 40         unconditional clock interupts       .......
                        test 41         resetting LKS                       .......     only done pass 1
                        test 42         line clock interrupts               yes         not CACHE dep, APT break sensitiv
e

                        test 43         maintenance register test           .......
                        test 44         serial line unit registers          .......
                        test 45         XCSR bit 7                          .......
                        test 46         RCSR bit 7 and XCSR bit 7           .......       done once in APT
                        test 47         RESET and XCSR<2!0>                 .......       first pass only

                        test 50         RESET and interrupt enable          .......
                        test 51         interrupt priority for SLU          .......       done once in APT
                        test 52         Break condition                     .......       done only once
                        test 53         overrun condition                   .......       done once in APT
                        test 54         LED'S on test                       .......         HOB on
                        test 55         MEMORY mapping    not done chain mode, APT, other than  pass 1
                        test 56         wrong parity abort test             .......
                        test 57         DMA TAG PARITY in standalone    yes                 HOB disabled

                        test 60         DMA tag parity w/o standalone   yes
                        test 61         DMA write hit cycles            yes
                        test 62         different levels of interrupts              not UFD, only if Q22BE found
                        test 63         Arbitration bet PIRQ interrupt              not UFD, only if Q22BE found
                        test 64         power down test                             not UFD, only if Q22BE found
                        test 65         arbitration between interrupt levels        not UFD, only if Q22BE found
                        test 66         PMG counter                     yes         not UFD, only if two Q22BE found

                        as of july 1984, this is the list of subtests, any added subtests will require
                        editing of this list....


                        note 1.  This test will only be run in first pass, and then only if bit 12 is set
                                 in the SWR.... Running this test will DESTROY everything in the EEROM
                                 except the first 109 bytes..... UFD area, secondary boots, and local
                                 language area's would all need to be restored after running this test.
                                 The first pass run time must be adjusted accordingly when running under
```

10. PROGRAM UPDATES AND MODIFICATIONS

1. CHANGES MADE BY HOWARD L. MARSHALL - DENOTED BY "$$$"
   IN COMMENT LINES OF ADDED OR CHANGED CODE, AS OF 2-22-85

ADDED SUBROUTINE "DETFPA" (DETERMINE FLOATING POINT ACCELERATOR)

THIS SUBROUTINE IS CALLED JUST PRIOR TO THE ERROR MACRO FOR ALL FLOATING POINT ERRORS. THIS SUBROUTINE PRINTS 1 OF 2 POSSIBLE ERROR MESSAGES. THEY ARE:

A.) ERROR DETECTED IN FLOATING POINT ACCELERATOR CHIP.

B.) ERROR DETECTED IN J11 FLOATING POINT PROCESSOR.

OF THE 2 ABOVE ERROR MESSAGES, THE ERROR MESSAGE THAT IS PRINTED IS DETERMINED BY THE STATE OF THE "FPA" FLAG, BIT 8, IN THE MAINTENANCE REGISTER. THIS BIT IS SET TO A 1 IF THE FLOATING POINT ACCELERATOR CHIP IS INSTALLED IN THE CPU BOARD. OTHERWISE, THE BIT IS CLEARED WHICH INDICATES THAT THE FLOATING POINT ACCELERTOR CHIP IS NOT INSTALLED. BASED SOLELY ON THIS DETERMINATION, IT IS LOGICAL TO ASSUME THAT MOST, IF NOT ALL FLOATING POINT ERRORS CAN BE ATTRIBUTED TO THE FPA CHIP IF THE "FPA" BIT IS SET OR TO THE FLOATING POINT PROCESSOR WITHIN THE J11 IF THE "FPA" BIT IS CLEARED.

```
                                .NLIST  BEX     ;DON'T EXPAND ALL THE ASCII.
167400                          $SWR=167400
000300                          $SWRMK=300
                        .TITLE  COKDADO KDJ11-B CLUSTER DIAG.
                        ;*COPYRIGHT (C) FEB 85
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY DIAG. ENG.
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C8), OCT, 1982.
                        ;*
000001                  $TN=1
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ------          --------------------
                        ;*        15            HALT ON ERROR
                        ;*        14            LOOP ON TEST
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*        11            INHIBIT ITERATIONS
                        ;*        10            BELL ON ERROR
                        ;*         9            LOOP ON ERROR
                        ;*         8            LOOP ON TEST IN SWR<5:0>
                        ;*         7            DO EXTENSIVE DATA RAM TEST
                        ;*         6            DO EXTENSIVE TAG RAM TEST
                        .SBTTL  MEMORY MANAGEMENT DEFINITIONS
                        ;*KT11 VECTOR ADDRESS
000250                  MMVEC=  250
                        ;*KT11 STATUS REGISTER ADDRESSES
177572                  SR0=    177572
177574                  SR1=    177574
177576                  SR2=    177576
172516                  SR3=    172516
                        ;*USER "I" PAGE DESCRIPTOR REGISTERS
177600                  UIPDR0= 177600
177602                  UIPDR1= 177602
177604                  UIPDR2= 177604
177606                  UIPDR3= 177606
177610                  UIPDR4= 177610
177612                  UIPDR5= 177612
177614                  UIPDR6= 177614
177616                  UIPDR7= 177616
                        ;*USER "D" PAGE DESCRIPTOR REGISTORS
177620                  UDPDR0= 177620
177622                  UDPDR1= 177622
177624                  UDPDR2= 177624
177626                  UDPDR3= 177626
177630                  UDPDR4= 177630
177632                  UDPDR5= 177632
177634                  UDPDR6= 177634
177636                  UDPDR7= 177636
                        ;*USER "I" PAGE ADDRESS REGISTERS
177640                  UIPAR0= 177640
177642                  UIPAR1= 177642
177644                  UIPAR2= 177644
```

```
                177646          UIPAR3= 177646
                177650          UIPAR4= 177650
                177652          UIPAR5= 177652
                177654          UIPAR6= 177654
                177656          UIPAR7= 177656
                                ;*USER "D" PAGE ADDRESS REGISTERS
                177660          UDPAR0= 177660
                177662          UDPAR1= 177662
                177664          UDPAR2= 177664
                177666          UDPAR3= 177666
                177670          UDPAR4= 177670
                177672          UDPAR5= 177672
                177674          UDPAR6= 177674
                177676          UDPAR7= 177676
                                ;*SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS
                172200          SIPDR0= 172200
                172202          SIPDR1= 172202
                172204          SIPDR2= 172204
                172206          SIPDR3= 172206
                172210          SIPDR4= 172210
                172212          SIPDR5= 172212
                172214          SIPDR6= 172214
                172216          SIPDR7= 172216
                                ;*SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS
                172220          SDPDR0= 172220
                172222          SDPDR1= 172222
                172224          SDPDR2= 172224
                172226          SDPDR3= 172226
                172230          SDPDR4= 172230
                172232          SDPDR5= 172232
                172234          SDPDR6= 172234
                172236          SDPDR7= 172236
                                ;*SUPERVISOR "I" PAGE ADDRESS REGISTERS
                172240          SIPAR0= 172240
                172242          SIPAR1= 172242
                172244          SIPAR2= 172244
                172246          SIPAR3= 172246
                172250          SIPAR4= 172250
                172252          SIPAR5= 172252
                172254          SIPAR6= 172254
                172256          SIPAR7= 172256
                                ;*SUPERVISOR "D" PAGE ADDRESS REGISTERS
                172260          SDPAR0= 172260
                172262          SDPAR1= 172262
                172264          SDPAR2= 172264
                172266          SDPAR3= 172266
                172270          SDPAR4= 172270
                172272          SDPAR5= 172272
                172274          SDPAR6= 172274
                172276          SDPAR7= 172276
                                ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
                172300          KIPDR0= 172300
                172302          KIPDR1= 172302
                172304          KIPDR2= 172304
                172306          KIPDR3= 172306
                172310          KIPDR4= 172310
                172312          KIPDR5= 172312
```

```
                172314          KIPDR6= 172314
                172316          KIPDR7= 172316
                                ;*KERNEL "D" PAGE DESCRIPTOR REGISTERS
                172320          KDPDR0= 172320
                172322          KDPDR1= 172322
                172324          KDPDR2= 172324
                172326          KDPDR3= 172326
                172330          KDPDR4= 172330
                172332          KDPDR5= 172332
                172334          KDPDR6= 172334
                172336          KDPDR7= 172336
                                ;*KERNEL "I" PAGE ADDRESS REGISTERS
                172340          KIPAR0= 172340
                172342          KIPAR1= 172342
                172344          KIPAR2= 172344
                172346          KIPAR3= 172346
                172350          KIPAR4= 172350
                172352          KIPAR5= 172352
                172354          KIPAR6= 172354
                172356          KIPAR7= 172356
                                ;*KERNEL "D" PAGE ADDRESS REGISTERS
                172360          KDPAR0= 172360
                172362          KDPAR1= 172362
                172364          KDPAR2= 172364
                172366          KDPAR3= 172366
                172370          KDPAR4= 172370
                172372          KDPAR5= 172372
                172374          KDPAR6= 172374
                172376          KDPAR7= 172376
474                             .SBTTL  BASIC DEFINITIONS
                                ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
                001100          STACK=  1100
                104000          ERROR=  EMT             ;;BASIC DEFINITION OF ERROR CALL
                000004          SCOPE=  IOT             ;;BASIC DEFINITION OF SCOPE CALL
                                ;*MISCELLANEOUS DEFINITIONS
                000011          HT=     11              ;;CODE FOR HORIZONTAL TAB
                000012          LF=     12              ;;CODE FOR LINE FEED
                000015          CR=     15              ;;CODE FOR CARRIAGE RETURN
                000200          CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
                177776          PS=     177776          ;;PROCESSOR STATUS WORD
                177776          PSW=    PS
                177774          STKLMT= 177774          ;;STACK LIMIT REGISTER
                177772          PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
                177570          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
                177570          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
                                ;*GENERAL PURPOSE REGISTER DEFINITIONS
                000000          R0=     %0              ;;GENERAL REGISTER
                000001          R1=     %1              ;;GENERAL REGISTER
                000002          R2=     %2              ;;GENERAL REGISTER
                000003          R3=     %3              ;;GENERAL REGISTER
                000004          R4=     %4              ;;GENERAL REGISTER
                000005          R5=     %5              ;;GENERAL REGISTER
                000006          R6=     %6              ;;GENERAL REGISTER
                000007          R7=     %7              ;;GENERAL REGISTER
                000006          SP=     %6              ;;STACK POINTER
                000007          PC=     %7              ;;PROGRAM COUNTER
                                ;*PRIORITY LEVEL DEFINITIONS
```

```
000000                  PR0=    0               ;;PRIORITY LEVEL 0
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
000140                  PR3=    140             ;;PRIORITY LEVEL 3
000200                  PR4=    200             ;;PRIORITY LEVEL 4
000240                  PR5=    240             ;;PRIORITY LEVEL 5
000300                  PR6=    300             ;;PRIORITY LEVEL 6
000340                  PR7=    340             ;;PRIORITY LEVEL 7
                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
001000                  SW9=    SW09
000400                  SW8=    SW08
000200                  SW7=    SW07
000100                  SW6=    SW06
000040                  SW5=    SW05
000020                  SW4=    SW04
000010                  SW3=    SW03
000004                  SW2=    SW02
000002                  SW1=    SW01
000001                  SW0=    SW00
                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00=  1
001000                  BIT9=   BIT09
000400                  BIT8=   BIT08
000200                  BIT7=   BIT07
000100                  BIT6=   BIT06
000040                  BIT5=   BIT05
```

```
                000020                  BIT4=   BIT04
                000010                  BIT3=   BIT03
                000004                  BIT2=   BIT02
                000002                  BIT1=   BIT01
                000001                  BIT0=   BIT00
                                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
                000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
                000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                000014                  TBITVEC=14              ;;"T" BIT
                000014                  TRTVEC= 14              ;;TRACE TRAP
                000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
                000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                000024                  PWRVEC= 24              ;;POWER FAIL
                000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
                000034                  TRAPVEC=34              ;;"TRAP" TRAP
                000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
                000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
                000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
    475         000001                          UFDSET=         1               ;FLAG FOR UFD
    476
                000000                  .SBTTL  TRAP CATCHER
                                                .=0
                                        ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                        ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                        ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                000174                  .=174
    000174      000000              DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
    000176      000000              SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
    477         000200              .=200
    478 000200  005037  001160              CLR     $TMP0
    479 000204  000137  004024              JMP     @#START
    480         000220              .=220
    481 000220  012737  000777  001160      MOV     #777,$TMP0
    482 000226  000137  004024              JMP     @#START
    483
    484                             .SBTTL  ACT11 HOOKS
                                    ;;**************************************************************
                                    ;HOOKS REQUIRED BY ACT11
                000232                  $SVPC=.                 ;SAVE PC
                000046                  .=46
    000046      140440                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                000052                  .=52
    000052      000000                  .WORD   0               ;;2)SET LOC.52 TO ZERO
                000232                  .=$SVPC                 ;; RESTORE PC
    485                             .SBTTL  APT PARAMETER BLOCK
                                    ;;**************************************************************
                                    ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                    ;;**************************************************************
                000232                  .$X=.   ;;SAVE CURRENT LOCATION
                000024                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
    000024      000200                  200     ;;FOR APT START UP
                000044                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
    000044      000232                  $APTHDR ;;POINT TO APT HEADER BLOCK
                000232                  .=.$X   ;;RESET LOCATION COUNTER
                                    ;;**************************************************************
                                    ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                    ;INTERFACE SPEC.
    000232                          $APTHD:
```

APT PARAMETER BLOCK

```
000232  000000                  $HIBTS: .WORD   0         ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
000234  001200                  $MBADR: .WORD   $MAIL     ;;ADDRESS OF APT MAILBOX (BITS 0-15)
000236  000000                  $TSTM:  .WORD             ;;RUN TIM OF LONGEST TEST
000240  000000                  $PASTM: .WORD             ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
000242  000000                  $UNITM: .WORD             ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
000244  000052                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
                                .SBTTL  COMMON TAGS
                                ;;*****************************************************************
                                ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                ;*USED IN THE PROGRAM.
                                        .=1100
001100                          $CMTAG:                         ;;START OF COMMON TAGS
001100  000000                          .WORD   0
001102  000                     $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103  000                     $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000                  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114  000                     $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115  001                     $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000                  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
001132  000000                          .WORD   0
001134  000                     $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
001135  000                     $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
001136  000000                          .WORD   0
001140  177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
001142  177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
001144  177560                  $TKS:   177560                  ;;TTY KBD STATUS
001146  177562                  $TKB:   177562                  ;;TTY KBD BUFFER
001150  177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
001154  000                     $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
001155  002                     $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156  012                     $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157  000                     $TPFLG: .BYTE   0               ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
000002                          .REPT   2
001160  000000                  $TMPO:  .WORD   0               ;;USER DEFINED
001162  000000                  $TMP1:  .WORD   0               ;;USER DEFINED
001164  000000                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001166  000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001170  207     377     377     $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001174  077                     $QUES:  .ASCII  /?/             ;;QUESTION MARK
001175  015                     $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001176  012     000             $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;*****************************************************************
                                .SBTTL  APT MAILBOX-ETABLE
                                ;;*****************************************************************
                                .EVEN
001200                          $MAIL:                          ;;APT MAILBOX
001200  000000                  $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
001202  000000                  $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
001204  000000                  $TESTN: .WORD   ATESTN          ;;TEST NUMBER
001206  000000                  $PASS:  .WORD   APASS           ;;PASS COUNT
001210  000000                  $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
001212  000000                  $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
001214  000000                  $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
001216  000000                  $MSGLG: .WORD   AMSGLG          ;;MESSAGE LENGTH
```

```
001220                          $ETABLE:                ;;APT ENVIRONMENT TABLE
001220     000                  $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
001221     000                  $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
001222  000000                  $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
001224  000000                  $USWR:  .WORD   AUSWR   ;;USER SWITCHES
001226  000000                  $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
                                ;*                      BITS 15-11=CPU TYPE
                                ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                ;*                              11/70=06,PDQ=07,Q=10
                                ;*                      BIT 10=REAL TIME CLOCK
                                ;*                      BIT  9=FLOATING POINT PROCESSOR
                                ;*                      BIT  8=MEMORY MANAGEMENT
001230     000                  $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
001231     000                  $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
                                ;*                              900 NSEC CORE=001
                                ;*                              300 NSEC BIPOLAR=002
                                ;*                              500 NSEC MOS=003
001232  000000                  $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001234     000                  $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
001235     000                  $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
001236  000000                  $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
001240     000                  $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
001241     000                  $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
001242  000000                  $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
001244     000                  $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
001245     000                  $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
001246  000000                  $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
001250  000000                  $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001252  000000                  $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001254  000000                  $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001256  000000                  $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
001260  000000                  $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
001262  000000                  $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
001264  000000                  $DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
001266  000000                  $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
001270  000000                  $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
001272  000000                  $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
001274  000000                  $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
001276  000000                  $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
001300  000000                  $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
001302  000000                  $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
001304  000000                  $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
001306  000000                  $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
001310  000000                  $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
001312  000000                  $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
001314  000000                  $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
001316  000000                  $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
001320  000000                  $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
001322  000000                  $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
001324                          $ETEND:
```

```
                                .SBTTL  ERROR POINTER TABLE
                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT
    001324                      $ERRTB:
 487
 488
 489                            .SBTTL  ERROR DEFINITIONS
 490                            ;ITEM 1
 491 001324  127002                     EM1             ;CPU ERROR
 492 001326  134621                     DH1             ;TEST #, ERROR PC
 493 001330  136026                     DT1             ;$TMP1,$ERRPC
 494 001332  000000                     0
 495                            ;ITEM 2
 496 001334  127036                     EM2             ;MMU ERROR
 497 001336  134621                     DH1             ;TEST #, ERROR PC
 498 001340  136026                     DT1             ;$TMP1,$ERRPC
 499 001342  000000                     0
 500                            ;ITEM 3
 501 001344  127050                     EM3             ;FPP ERROR
 502 001346  134621                     DH1             ;TEST #, ERROR PC
 503 001350  136026                     DT1             ;$TMP1,$ERRPC
 504 001352  000000                     0
 505                            ;ITEM 4
 506 001354  127062                     EM4             ;ERROR IN READ-WRITE BITS OF CCR
 507 001356  134646                     DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
 508 001360  136034                     DT4             ;$TMP1,$ERRPC,R1,CCR
 509 001362  000000                     0
 510                            ;ITEM 5
 511 001364  127122                     EM5             ;FORCE MISS WRITES TO CACHE
 512 001366  134733                     DH5             ;TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
 513 001370  136046                     DT5             ;$TMP1,$ERRPC, R2, R1, $GDDAT
 514 001372  000000                     0
 515                            ;ITEM 6
 516 001374  127155                     EM6             ;FORCE MISS WRITE INVALIDATES CACHE
 517 001376  134733                     DH5             ;TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
 518 001400  136046                     DT5             ;$TMP1,$ERRPC, R2, R1, $GDDAT
 519 001402  000000                     0
 520                            ;ITEM 7
 521 001404  127220                     EM7             ;UNEXPECTED PARITY INTERRUPT
 522 001406  135032                     DH7             ;TEST #, PC, ADDRESS ACCESSED, MSER
 523 001410  136062                     DT7             ;$TMP1,$ERRPC,$BDADR,MSER
 524 001412  000000                     0
 525                            ;ITEM 10
 526 001414  127254                     EM10            ;TAG PARITY ERROR
 527 001416  135032                     DH7             ;TEST #, PC, ADDRESS ACCESSED, MSER
 528 001420  136062                     DT7             ;$TMP1,$ERRPC,$BDADR,MSER
 529 001422  000000                     0
 530                            ;ITEM 11
 531 001424  127275                     EM11            ;DATA PARITY ERROR
 532 001426  135032                     DH7             ;TEST #, PC, ADDRESS ACCESSED, MSER
```

ERROR DEFINITIONS

```
533 001430  136062                      DT7             ;$TMP1,$ERRPC,$BDADR,MSER
534 001432  000000                      0
535                             ;ITEM 12
536 001434  127317                      EM12            ;LOW BYTE PARITY ERROR
537 001436  135032                      DH7             ;TEST #, PC, ADDRESS ACCESSED, MSER
538 001440  136062                      DT7             ;$TMP1,$ERRPC,$BDADR,MSER
539 001442  000000                      0
540                             ;ITEM 13
541 001444  127345                      EM13            ;HIGH BYTE PARITY ERROR
542 001446  135032                      DH7             ;TEST #, PC, ADDRESS ACCESSED, MSER
543 001450  136062                      DT7             ;$TMP1,$ERRPC,$BDADR,MSER
544 001452  000000                      0
545                             ;ITEM 14
546 001454  127374                      EM14            ;ERROR DATA PATH
547 001456  134646                      DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
548 001460  136074                      DT14            ;$TMP1,$ERRPC,$GDDAT,TSTLOC
549 001462  000000                      0
550                             ;ITEM 15
551 001464  127417                      EM15            ;FORCE MISS READS FROM CACHE
552 001466  134733                      DH5             ;TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
553 001470  136046                      DT5             ;$TMP1,$ERRPC, R2, R1, $GDDAT
554 001472  000000                      0
555                             ;ITEM 16
556 001474  127453                      EM16            ;FORCE MISS READS FROM CACHE AND MISS
557 001476  134733                      DH5             ;TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
558 001500  136046                      DT5             ;$TMP1,$ERRPC, R2, R1, $GDDAT
559 001502  000000                      0
560                             ;ITEM 17
561 001504  127520                      EM17            ;ERROR IN RECORDING HITS IN HIT/MISS
562 001506  134646                      DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
563 001510  136106                      DT17            ;$TMP1,$ERRPC,$GDDAT,RECDAT
564 001512  000000                      0
565                             ;ITEM 20
566 001514  127564                      EM20            ;WRITE BYTE ALLOCATES CACHE
567 001516  134621                      DH1             ;TEST #, PC
568 001520  136026                      DT1             ;$TMP1,$ERRPC
569 001522  000000                      0
570                             ;ITEM 21
571 001524  127617                      EM21            ;WRITE BYTE HIT DOES NOT RECORD HIT
572 001526  134621                      DH1             ;TEST #, PC
573 001530  136026                      DT1             ;$TMP1,$ERRPC
574 001532  000000                      0
575                             ;ITEM 22
576 001534  127662                      EM22            ;BYTES REVERSED ON WRITE CYCLES
577 001536  134621                      DH1             ;TEST #, PC
578 001540  136026                      DT1             ;$TMP1,$ERRPC
579 001542  000000                      0
580                             ;ITEM 23
581 001544  127721                      EM23            ;CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE
582 001546  134733                      DH5             ;TEST #, PC, HIT/MISS, DATA IN CACHE, DATA IN MEMORY
583 001550  136046                      DT5             ;$TMP1,$ERRPC, R2, R1, $GDDAT
584 001552  000000                      0
585                             ;ITEM 24
586 001554  127775                      EM24            ;HITS RECORDED AFTER FLUSHING CACHE
587 001556  135105                      DH24            ;TEST #, PC, NUMBER OF HITS
588 001560  136120                      DT24            ;$TMP1,$ERRPC,R3
589 001562  000000                      0
```

ERROR DEFINITIONS

```
                                ;ITEM 25
                                        EM25            ;BYPASS DOESN'T INVALIDATE CACHE
                                        DH1             ;TEST #, PC
                                        DT1             ;$TMP1,$ERRPC
                                        0
                                ;ITEM 26
                                        EM26            ;MSER DOES NOT CLEAR ON WRITE REFERENCE
                                        DH1             ;TEST #, PC
                                        DT1             ;$TMP1,$ERRPC
                                        0
                                ;ITEM 27
                                        EM27            ;PARITY ERROR DON'T CAUSE A MISS
                                        DH27            ;TEST #, PC, MSER, HIT/MISS
                                        DT27            ;$TMP1,$ERRPC,MSER,R3,0
                                        0
                                ;ITEM 30
                                        EM30            ;PARITY ERROR DON'T SET MSER WITH CCR<7>=0
                                        DH27            ;TEST #, PC, MSER, HIT/MISS
                                        DT27            ;$TMP1,$ERRPC,MSER,R3,0
                                        0
                                ;ITEM 31
                                        EM31            ;PARITY ERROR IGNORED
                                        DH1             ;TEST #, PC
                                        DT1             ;$TMP1,$ERRPC
                                        0
                                ;ITEM 32
                                        EM32            ;PARITY ERROR IGNORED ON LOW BYTE
                                        DH1             ;TEST #, PC
                                        DT1             ;$TMP1,$ERRPC
                                        0
                                ;ITEM 33
                                        EM33            ;PARITY ERROR IGNORED ON HIGH BYTE
                                        DH1             ;TEST #, PC
                                        DT1             ;$TMP1,$ERRPC
                                        0
                                ;ITEM 34
                                        EM34            ;PARITY ABORT LOGIC DOESN'T WORK
                                        DH1             ;TEST #, PC
                                        DT1             ;$TMP1,$ERRPC
                                        0
                                ;ITEM 35
                                        EM35            ;MSER NOT SET PROPERLY
                                        DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
                                        DT35            ;$TMP1,$ERRPC,$GDDAT,MSER,0
                                        0
                                ;ITEM 36
                                        EM36            ;PARITY INTERRUPT DOESN'T WORK
                                        DH1             ;TEST #, PC
                                        DT1             ;$TMP1,$ERRPC
                                        0
                                ;ITEM 37
                                        EM37            ;NXM AND PARITY ABORT DIN'T HAPPEN
                                        DH1             ;TEST #, PC
                                        DT1             ;$TMP1,$ERRPC
                                        0
                                ;ITEM 40
                                        EM40            ;PARITY ABORT NOT BLOCKED BY NXM TRAP
```

ERROR DEFINITIONS

```
647 001716  134621                      DH1             ;TEST #, PC
648 001720  136026                      DT1             ;$TMP1,$ERRPC
649 001722  000000                      0
650                             ;ITEM 41
651 001724  130652                      EM41            ;MULTI-PROCESSOR HOOK INSTRUCTION DOESN'T CAUSE MISS
652 001726  135214                      DH41            ;TEST #, PC, INSTRUCTION OPCODE
653 001730  136154                      DT41            ;$TMP1,$ERRPC,$BDDAT
654 001732  000000                      0
655                             ;ITEM 42
656 001734  130736                      EM42            ;ERROR IN PARITY LOGIC
657 001736  134621                      DH1             ;TEST #, PC
658 001740  136026                      DT1             ;$TMP1,$ERRPC
659 001742  000000                      0
660                             ;ITEM 43
661 001744  130764                      EM43            ;ERROR IN CACHE DATA RAMS
662 001746  135265                      DH43            ;TEST #, PC, EXPECTED DATA, RECEIVED DATA, CACHE LOCATION
663 001750  136164                      DT43            ;$TMP1,$ERRPC,R1,RECDAT,$BDADR
664 001752  000000                      0
665                             ;ITEM 44
666 001754  131015                      EM44            ;ERROR IN NXM IN STANDALONE MODE
667 001756  134621                      DH1             ;TEST #, PC
668 001760  136026                      DT1             ;$TMP1,$ERRPC
669 001762  000000                      0
670                             ;ITEM 45
671 001764  131055                      EM45            ;HITS NOT RECORDED PROPERLY THRU HIT/MISS
672 001766  134621                      DH1             ;TEST #, PC
673 001770  136026                      DT1             ;$TMP1,$ERRPC
674 001772  000000                      0
675                             ;ITEM 46
676 001774  131137                      EM46            ;HIT RECORDED FOR A LOCATION NOT IN CACHE
677 001776  135365                      DH47            ;TEST #, PC, LOCATION ACCESSED
678 002000  136200                      DT47            ;$TMP1,$ERRPC,KIPAR6,$BDADR
679 002002  000000                      0
680                             ;ITEM 47
681 002004  131227                      EM47            ;MISS RECORDED FOR A LOCATION THAT SHOULD BE IN CACHE
682 002006  135365                      DH47            ;TEST #, PC, LOCATION ACCESSED
683 002010  136200                      DT47            ;$TMP1,$ERRPC,KIPAR6,$BDADR
684 002012  000000                      0
685                             ;ITEM 50
686 002014  131314                      EM50            ;ERROR IN TAG STORE
687 002016  135365                      DH47            ;TEST #, PC, ADDRESS
688 002020  136212                      DT50            ;$TMP1,$ERRPC,R1,$BDADR
689 002022  000000                      0
690                             ;ITEM 51
691 002024  131337                      EM51            ;ERROR PCR READ-WRITE BITS
692 002026  134646                      DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
693 002030  136224                      DT51            ;$TMP1,$ERRPC,$GDDAT,PCR
694 002032  000000                      0
695                             ;ITEM 52
696 002034  131371                      EM52            ;ERROR IN BCSR READ-WRITE BITS
697 002036  134646                      DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
698 002040  136236                      DT52            ;$TMP1,$ERROC,$GDDAT,BCSR
699 002042  000000                      0
700                             ;ITEM 53
701 002044  131427                      EM53            ;RESET DOESN'T CLEAR BCSR<4>
702 002046  134621                      DH1             ;TEST #, PC
703 002050  136026                      DT1             ;$TMP1,$ERRPC
```

ERROR DEFINITIONS

```
    704 002052  000000                     0
    705                             ;ITEM 54
    706 002054  131463                      EM54            ;CHECKSUM ERROR IN 16-BIT ROM
    707 002056  134621                      DH1             ;TEST #, PC
    708 002060  136026                      DT1             ;$TMP1,$ERRPC
    709 002062  000000                      0
    710                             ;ITEM 55
    711 002064  131521                      EM55            ;CHCKSUM ERROR IN 8-BIT ROM
    712 002066  134621                      DH1             ;TEST #, PC
    713 002070  136026                      DT1             ;$TMP1,$ERRPC
    714 002072  000000                      0
    715                             ;ITEM 56
    716 002074  131555                      EM56            ;TIMEOUT READING LKS
    717 002076  134621                      DH1             ;TEST #, PC
    718 002100  136026                      DT1             ;$TMP1,$ERRPC
    719 002102  000000                      0
    720                             ;ITEM 57
    721 002104  131601                      EM57            ;LKS<07> DOES NOT BECOME 1
    722 002106  134621                      DH1             ;TEST #, PC
    723 002110  136026                      DT1             ;$TMP1,$ERRPC
    724 002112  000000                      0
    725                             ;ITEM 60
    726 002114  131633                      EM60            ;WRITE REFERENCE DOESN'T CLEAR LKS<07>
    727 002116  134621                      DH1             ;TEST #, PC
    728 002120  136026                      DT1             ;$TMP1,$ERRPC
    729 002122  000000                      0
    730                             ;ITEM 61
    731 002124  131701                      EM61            ;ILLEGAL LKS INTERRUPTS
    732 002126  134621                      DH1             ;TEST #, PC
    733 002130  136026                      DT1             ;$TMP1,$ERRPC
    734 002132  000000                      0
    735                             ;ITEM 62
    736 002134  131730                      EM62            ;PROCESSOR INTERRUPTS DON'T CLEAR LKS<07>
    737 002136  134621                      DH1             ;TEST #, PC
    738 002140  136026                      DT1             ;$TMP1,$ERRPC
    739 002142  000000                      0
    740                             ;ITEM 63
    741 002144  132001                      EM63            ;LKS READY DOESN'T GO LOW
    742 002146  134621                      DH1             ;TEST #, PC
    743 002150  136026                      DT1             ;$TMP1,$ERRPC
    744 002152  000000                      0
    745                             ;ITEM 64
    746 002154  132032                      EM64            ;WRONG NUMBER OF LKS INTERRUPTS
    747 002156  134621                      DH1             ;TEST #, PC
    748 002160  136026                      DT1             ;$TMP1,$ERRPC
    749 002162  000000                      0
    750                             ;ITEM 65
    751 002164  132071                      EM65            ;LKS INTERRUPTS HAPPEN AT WRONG PRIORITY
    752 002166  135452                      DH65            ;TEST #, PC, PRIORITY
    753 002170  136262                      DT65            ;$TMP1,$ERRPC,$GDDAT
    754 002172  000000                      0
    755                             ;ITEM 66
    756 002174  132141                      EM66            ;BCSR<12> DOES NOT DISABLES LKS
    757 002176  134621                      DH1             ;TEST #, PC
    758 002200  136026                      DT1             ;$TMP1,$ERRPC
    759 002202  000000                      0
    760                             ;ITEM 67
```

ERROR DEFINITIONS

```
761 002204  132200                  EM67            ;BCSR<13> DOESN'T SET LKS<06>
762 002206  134621                  DH1             ;TEST #, PC
763 002210  136026                  DT1             ;$TMP1,$ERRPC
764 002212  000000                  0
765                         ;ITEM 70
766 002214  132235                  EM70            ;RESET DOESN'T SET LKS<7>
767 002216  134621                  DH1             ;TEST #, PC
768 002220  136026                  DT1             ;$TMP1,$ERRPC
769 002222  000000                  0
770                         ;ITEM 71
771 002224  132266                  EM71            ;RESET DOESN'T CLEAR LKS<06>
772 002226  134621                  DH1             ;TEST #, PC
773 002230  136026                  DT1             ;$TMP1,$ERRPC
774 002232  000000                  0
775                         ;ITEM 72
776 002234  132322                  EM72            ;TIMEOUT READING SLU REGISTERS
777 002236  135516                  DH72            ;TEST #, PC, ADDRESS FAILED
778 002240  136154                  DT41            ;$TMP1,$ERRPC,$BDDAT
779 002242  000000                  0
780                         ;ITEM 73
781 002244  132360                  EM73            ;XMIT READY DIDN'T GO LOW
782 002246  134621                  DH1             ;TEST #, PC
783 002250  136026                  DT1             ;$TMP1,$ERRPC
784 002252  000000                  0
785                         ;ITEM 74
786 002254  132404                  EM74            ;RCSR DOESN'T BECOME 1
787 002256  134621                  DH1             ;TEST #, PC
788 002260  136026                  DT1             ;$TMP1,$ERRPC
789 002262  000000                  0
790                         ;ITEM 75
791 002264  132435                  EM75            ;WRONG CHARACTER RECEIVED
792 002266  134646                  DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
793 002270  136272                  DT75            ;$TMP1,$ERRPC,$GDDAT,$BDDAT
794 002272  000000                  0
795                         ;ITEM 76
796 002274  132466                  EM76            ;RCSR<07> NOT CLEARED AFTER READING RBUF
797 002276  134621                  DH1             ;TEST #, PC
798 002300  136026                  DT1             ;$TMP1,$ERRPC
799 002302  000000                  0
800                         ;ITEM 77
801 002304  132536                  EM77            ;XCSR<07> NOT SET ON RESET
802 002306  134621                  DH1             ;TEST #, PC
803 002310  136026                  DT1             ;$TMP1,$ERRPC
804 002312  000000                  0
805                         ;ITEM 100
806 002314  132570                  EM100           ;RCSR<07> NOT CLEARED ON RESET
807 002316  134621                  DH1             ;TEST #, PC
808 002320  136026                  DT1             ;$TMP1,$ERRPC
809 002322  000000                  0
810                         ;ITEM 101
811 002324  132626                  EM101           ;SLU INTERRUPTS HAPPEN AT 4
812 002326  134621                  DH1             ;TEST #, PC
813 002330  136026                  DT1             ;$TMP1,$ERRPC
814 002332  000000                  0
815                         ;ITEM 102
816 002334  132661                  EM102           ;RESET DOES NOT CLEAR XCSR<6> AND RSCR<6>
817 002336  134621                  DH1             ;TEST #, PC
```

ERROR DEFINITIONS

```
818 002340  136026                      DT1             ;$TMP1,$ERRPC
819 002342  000000                      0
820                             ;ITEM 103
821 002344  132743                      EM103           ;TRANSMIT INTERRUPT DOES NOT CLEAR XCSR<07>
822 002346  134621                      DH1             ;TEST #, PC
823 002350  136026                      DT1             ;$TMP1,$ERRPC
824 002352  000000                      0
825                             ;ITEM 104
826 002354  133016                      EM104           ;RECEIVE INTERRUPTS DON'T CLEAR RCSR<07>
827 002356  134621                      DH1             ;TEST #, PC
828 002360  136026                      DT1             ;$TMP1,$ERRPC
829 002362  000000                      0
830                             ;ITEM 105
831 002364  133066                      EM105           ;BREAK CONDITION DOES NOT SET RBUF PROPERLY
832 002366  135562                      DH105           ;TEST #, PC, RBUF
833 002370  136304                      DT105           ;$TMP1,$ERRPC,RBUF
834 002372  000000                      0
835                             ;ITEM 106
836 002374  133141                      EM106           ;RBUF WASN'T CLEARED ON NEXT CHAR.
837 002376  135562                      DH105           ;TEST #, PC, RBUF
838 002400  136304                      DT105           ;$TMP1,$ERRPC,RBUF
839 002402  000000                      0
840                             ;ITEM 107
841 002404  133217                      EM107           ;ERROR IN WRITING TO XCSR<0>
842 002406  134621                      DH1             ;TEST #, PC
843 002410  136026                      DT1             ;$TMP1,$ERRPC
844 002412  000000                      0
845                             ;ITEM 110
846 002414  133253                      EM110           ;RESET DOES NOT CLEAR XCSR<00>
847 002416  134621                      DH1             ;TEST #, PC
848 002420  136026                      DT1             ;$TMP1,$ERRPC
849 002422  000000                      0
850                             ;ITEM 111
851 002424  133315                      EM111           ;FIRST CHARACTER WAS NOT OVERRUN BY THE SECOND
852 002426  134646                      DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
853 002430  136272                      DT75            ;$TMP1,$ERRPC,$GDDAT,$BDDAT
854 002432  000000                      0
855                             ;ITEM 112
856 002434  133373                      EM112           ;OVERRUN CONDITION DOES NOT SET PROPER BITS IN RBUF
857 002436  135562                      DH105           ;TEST #, PC, RBUF
858 002440  136304                      DT105           ;$TMP1,$ERRPC,RBUF
859 002442  000000                      0
860                             ;ITEM 113
861 002444  133456                      EM113           ;RBUF WAS NOT CLEARED ON THE NEXT CHARACTER
862 002446  135562                      DH105           ;TEST #, PC, RBUF
863 002450  136304                      DT105           ;$TMP1,$ERRPC,RBUF
864 002452  000000                      0
865                             ;ITEM 114
866 002454  133542                      EM114           ;ERROR IN XCSR<2>
867 002456  134621                      DH1             ;TEST #, PC
868 002460  136026                      DT1             ;$TMP1,$ERRPC
869 002462  000000                      0
870                             ;ITEM 115
871 002464  133563                      EM115           ;ERROR IN TAG STORE FROM STANDALONE MODE
872 002466  135614                      DH115           ;TEST #, PC, MSER, ADDRESS ACCESSED
873 002470  136314                      DT115           ;$TMP1,$ERRPC,$BDDAT,KIPAR6,$BDADR
874 002472  000000                      0
```

ERROR DEFINITIONS

```
875
876 002474  133633                ;ITEM 116
877 002476  134621                        EM116           ;DMA TAG PARITY DOES NOT SET MSER<4>
878 002500  136026                        DH1             ;TEST #, PC
879 002502  000000                        DT1             ;$TMP1,$ERRPC
880                                       0
881 002504  133714                ;ITEM 117
882 002506  134621                        EM117           ;MSER<13>NOT SET IN STANDALONE MODE
883 002510  136026                        DH1             ;TEST #, PC
884 002512  000000                        DT1             ;$TMP1,$ERRPC
885                                       0
886 002514  133757                ;ITEM 120
887 002516  134621                        EM120           ;DMA WRITE HITS DON'T INVALIDATE CACHE
888 002520  136026                        DH1             ;TEST #, PC
889 002522  000000                        DT1             ;$TMP1,$ERRPC
890                                       0
891 002524  134025                ;ITEM 121
892 002526  134621                        EM121           ;IN BLOCK MODE ON WRITE DMA HITS NOT EVERYTHING IS INVALIDATED
893 002530  136026                        DH1             ;TEST #, PC
894 002532  000000                        DT1             ;$TMP1,$ERRPC
895                                       0
896 002534  134123                ;ITEM 122
897 002536  134621                        EM122           ;READ DMA HIT IS MESSED UP
898 002540  136026                        DH1             ;TEST #, PC
899 002542  000000                        DT1             ;$TMP1,$ERRPC
900                                       0
901 002544  134151                ;ITEM 123
902 002546  134621                        EM123           ;ERROR IN DMA CYCLES FROM Q22BE
903 002550  136026                        DH1             ;TEST #, PC
904 002552  000000                        DT1             ;$TMP1,$ERRPC
905                                       0
906 002554  134203                ;ITEM 124
907 002556  134621                        EM124           ;PIRQ INTERRUPTS DON'T TAKE PRIORITY OVER Q BUS INTERRUPTS
908 002560  136026                        DH1             ;TEST #, PC
909 002562  000000                        DT1             ;$TMP1,$ERRPC
910                                       0
911 002564  134275                ;ITEM 125
912 002566  134621                        EM125           ;NO POWER DOWN TRAP TO 24 OCCUR
913 002570  136026                        DH1             ;TEST #, PC
914 002572  000000                        DT1             ;$TMP1,$ERRPC
915                                       0
916 002574  134334                ;ITEM 126
917 002576  134621                        EM126           ;ERROR IN INTERRUPTS FROM Q22BE
918 002600  136026                        DH1             ;TEST #, PC
919 002602  000000                        DT1             ;$TMP1,$ERRPC
920                                       0
921 002604  134371                ;ITEM 127
922 002606  134621                        EM127           ;ERROR IN PMG COUNTER
923 002610  136026                        DH1             ;TEST #, PC
924 002612  000000                        DT1             ;$TMP1,$ERRPC
925                                       0
926 002614  134433                ;ITEM 130
927 002616  134621                        EM130           ;UNEXPECTED TIMEOUT
928 002620  136330                        DH1             ;TEST #, PC
929 002622  000000                        DT130           ;$TMP1,$ERRPC
930                                       0
931 002624  134460                ;ITEM 131
                                          EM131           ;ERROR WRITING TO LKS<6>
```

ERROR DEFINITIONS

```
                                        DH1             ;TEST #, PC
                                        DT1             ;$TMP1,$ERRPC
                                        0
                                ;ITEM 132
                                        EM132           ;MAINTENANCE REGISTER ERROR
                                        DH1             ;TEST #, PC
                                        DT1             ;$TMP1,$ERRPC
                                        0
                                ;ITEM 133
                                        EM133           ; EEROM TYPE ERROR
                                        DH1             ;TEST #, PC
                                        DT1             ;$TMP1,$ERRPC
                                        0
                                ;ITEM 134
                                        EM134           ; ERROM WRITE/READ ERROR
                                        DH134           ;TEST #, PC, ERROR COUNT, PATTERN, DATA READ, ADDRESS
                                        DT134           ;$TMP1,$ERRPC,$TMP0,R3,$BDDAT,$BDADR
                                        0

                                .SBTTL  GLOBAL VARIABLES AND REGISTER NAMES

                                ;REGISTERS FOR THE FIRST Q22BE
                                CSR1:   .WORD   0                       ;CONTROL REGISTER 1 FOR Q22BE
                                CSR2:   .WORD   0                       ;CONTROL/STATUS REGISTER 2
                                BA:     .WORD   0                       ;DMA ADDRESS FOR Q22BE
                                WC:     .WORD   0                       ;WORD COUNT REGISTER
                                DATA:   .WORD   0                       ;DMA DATA FOR Q22BE
                                VQBE1:  .WORD   0                       ;ADDRESS OF VECTOR FOR Q22BE
                                VQPR1:  .WORD   0                       ;PRIORITY
                                SIMGOA: .WORD   0                       ;SIMULTANEUOS GO ADDRESS REGISTER

                                ;REGISTERS FOR THE SECOND Q22BE
                                CSR12:  .WORD   0                       ;CONTROL REGISTER 1 FOR Q22BE
                                CSR22:  .WORD   0                       ;CONTROL/STATUS REGISTER 2
                                BA2:    .WORD   0                       ;DMA ADDRESS FOR Q22BE
                                WC2:    .WORD   0                       ;WORD COUNT REGISTER
                                DATA2:  .WORD   0                       ;DMA DATA FOR Q22BE
                                VQBE2:  .WORD   0                       ;ADDRESS OF VECTOR FOR Q22BE
                                VQPR2:  .WORD   0                       ;PRIORITY

                                LKSFL:  .WORD   0
                                ACTCHS: .WORD   0                       ;ACTUAL CHECKSUM
                                SAVPCR: .WORD   0
                                SAVBR:  .WORD   0
                        002740  .=2740
                                TEMP:   .WORD   0
                                        .BLKW   15.                     ;RESERVED FOR BLOCK MODE TRANSFER
                                TIMOUT: .WORD   0
                                Q22EN:  .WORD   32,12,6,2               ;PRIORITY 7-4 FOR Q22BE


                        177524                  BCR=            177524  ;BOOT/DIAGNOSTICS CONFIGURATION
                        177524                  BDR=            177524  ;BOOT/DIAGNOSTICS DISPLAY
                        177520                  BCSR=           177520  ;BOOT/DIAGNOSTICS STATUS
                        177746                  CCR=            177746  ;CACHE CONTROL REGISTER
                        177752                  HITMIS=         177752  ;HIT OR MISS REGISTER
                        177734                  KMCR=           177734  ;UNIBUS CONFIGURATION REGISTER
```

GLOBAL VARIABLES AND REGISTER NAMES

```
 989          177546                          LKS=            177546          ;CLOCK STATUS REGISTER
 990          177750                          MAIREG=         177750          ;MAINTENANCE REGISTER
 991          177744                          MSER=           177744          ;MEMORY SYSTEM ERROR
 992          177522                          PCR=            177522          ;PAGE CONTROL REGISTER
 993          177772                          PIR=            177772          ;PROGRAM INTERRUPT REQUEST
 994          177562                          RBUF=           177562          ;RECEIVER DATA BUFFER
 995          177560                          RCSR=           177560          ;RECEIVER STATUS REGISTER
 996          177566                          XBUF=           177566          ;TRANSMITTER DATA BUFFER
 997          177564                          XCSR=           177564          ;TRANSMITTER STATUS REGISTER
 998
 999          177766                          CPEREG=         177766          ;CPU ERROR REGISTER
1000
1001          177572                          MMR0=SR0                        ;MEMORY MANAGEMENT REG. 0
1002          177574                          MMR1=SR1                        ;MEMORY MANAGEMENT REG. 1
1003          177576                          MMR2=SR2                        ;MEMORY MANAGEMENT REG. 2
1004          172516                          MMR3=SR3                        ;MEMORY MANAGEMENT REG. 3
1005          120001                          POLY= 120001
1006          000000                          NULL=           0
1007
1008                                  .SBTTL  GLOBAL DATA SECTION
1009
1010                                  ;++
1011                                  ; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
1012                                  ; IN MORE THAN ONE TEST.
1013                                  ;--
1014
1015                                  ;THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE VECTOR DATA
1016                                  ;WHEN THE TEST NEEDS TO HAVE AN ERROR CONDITION RESPOND DIFFERENTLY
1017                                  ;FROM THE DEFAULT RESPONCE.
1018 003012  000000                   SLOCO0: .WORD   0
1019 003014  000000                   SLOCO1: .WORD   0
1020
1021                                  ;THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE WORKING DATA.
1022 003016  000000                   LOWADD: .WORD   0               ;STORES LOW ADDRESS FOR RAM TESTS
1023 003020  000000                   GOODAD: .WORD   0               ;STORES GOOD ADDRESS FOR RAM TESTS
1024 003022  000000                   ERRCNT: .WORD   0               ; CONTAINS TOTAL NO. OF EEROM ERRORS
1025 003024  000000                   TSTADD: .WORD   0               ;ADDRESS STORE FOR RAM TESTS
1026 003026  000000                   NEWADD: .WORD   0               ;ADDRESS STORE FOR RAM TEST
1027 003030  000000                   FLAG:   .WORD   0               ;USED TO STORE "FLAG" CONDITIONS
1028 003032  000000                   CCHPAS: .WORD   0               ; flag-counter for control of Cache subtests
1029 003034  000000                   EEPAS:  .WORD   0               ; flag-counter for control of EEROM subtest
1030 003036  000000                   SAVSUP: .WORD   0               ;USED TO STORE SUPERVISOR STACK VALUE
1031 003040  000000                   SAVUSE: .WORD   0               ;USED TO STORE USER STACK VALUE
1032 003042  000000                   SAVMR0: .WORD   0               ;USED TO STORE MMU STATUS REGISTER 0 DATA
1033 003044  000000                   SAVMR1: .WORD   0               ;USED TO STORE MMU STATUS REGISTER 1 DATA
1034 003046  000000                   SAVMR2: .WORD   0               ;USED TO STORE MMU STATUS REGISTER 2 DATA
1035 003050  000000                   SAVSWR: .WORD   0               ; SAVE SFTWRE SWTCH REG DURING EEROM TEST
1036 003052                           FLOAT:  .BLKW   4               ;USED TO STORE VALUES FOR MMU TESTS
1037 003062                           FLO:    .BLKW   4               ;USED TO STORE VALUES FOR MMU TESTS
1038 003072  000000                   SEQ:    .WORD   0               ;STORES SEQUENCE NUMBER FOR JUMP TESTS
1039 003074  000000                   SPS:    .WORD   0               ;STORES STACK POINTER FOR JUMP TESTS
1040 003076  000000                   SPSJ:   .WORD   0               ;STORES STACK POINTER FOR JUMP TESTS
1041 003100                           BTEXP:  .BLKW   4               ;STORES EXPONENT DURING BIT TESTS
1042 003110                           BTRES:  .BLKW   4               ;STORES RECIEVED DATA FOR BIT TESTS
1043 003120  000000                   COUNT:  .WORD   0               ;ERROR INDICATOR FOR FLOATING POINT TESTS
1044 003122                           RECFEC: .BLKW   4               ;RECIEVED FLOATING POINT EXCEPTION CODE
1045 003132                           RECST:  .BLKW   4               ;RECIEVED FLOATING POINT STATUS
```

```
1046 003142                          RECDST: .BLKW   4               ;DESTINATION ADDRESS FOR FLOATING POINT TESTS
1047
1048                                 ;THESE LOCATIONS ARE USED BY MORE THAN ONE TEST AS LOOP COUNTERS
1049 003152  000000                  ALLCTR: .WORD   0
1050 003154  000000                  LOOPIN: .WORD   0
1051
1052                                 ;SOME MORE TEMPORARY STORAGE FOR RAM TESTS
1053 003156  000000                  SAVPOS: .WORD   0               ;STORES TEMPORARY BIT POSITIONS FOR RAM TESTS
1054 003160  000000                  MASK:   .WORD   0               ;STORES BIT MASK FOR ERROR ISOLATION
1055
1056                                 ;!!!!!THIS IS IT. THE PROGRAM TEST LOCATION!!!!!!!!!!!!!!!!!!
1057 003162  000000                  TSTLOC: .WORD   0
1058 003164                                  .BLKW   20.
1059                                         ;FPP REGISTER DEFINITIONS
1060         000000                          AC0= %0
1061         000001                          AC1= %1
1062         000002                          AC2= %2
1063         000003                          AC3= %3
1064         000004                          AC4= %4
1065         000005                          AC5= %5
1066         000006                          AC6= %6
1067         000007                          AC7= %7
1068
1069                                         ;FPP INTERRUPT VECTOR
1070
1071         000244                          FPVEC=244
1072
1073
1074         001000                          STBOT=  1000
1075
1076
1078 003234  123456                  TAB1:   .WORD   123456
1079 003236  000000                          .WORD   000000
1080 003240  000000                          .WORD   0
1081 003242  000001                          .WORD   1
1082 003244  055555                  TAB2:   .WORD   055555
1083 003246  177777                          .WORD   177777
1084 003250  145671                          .WORD   145671
1085 003252  100000                          .WORD   100000
1086 003254  003000                  TAB3:   .WORD   003000
1087 003256  123456                          .WORD   123456
1088 003260  000000                          .WORD   0
1089 003262  000000                          .WORD   0
1090 003264  055555                  TAB4:   .WORD   55555
1091 003266  177777                          .WORD   -1
1092 003270  000000                          .WORD   0
1093 003272  000000                          .WORD   0
1094 003274  043243                  TAB5:   .WORD   43243
1095 003276  000000                          .WORD   0
1096 003300  000000                          .WORD   0
1097 003302  000000                          .WORD   0
1098 003304  162400                  TAB5A:  .WORD   162400
1099 003306  000000                          .WORD   0
1100 003310  000000                          .WORD   0
1101 003312  000000                          .WORD   0
1102 003314  000000                  TAB6:   .WORD   0
1103 003316  000000                          .WORD   0
```

GLOBAL DATA SECTION

```
1104 003320  000000                          .WORD   0
1105 003322  000000                          .WORD   0
1106 003324  047050                TAB6A:    .WORD   47050
1107 003326  010000                          .WORD   10000
1108 003330  000000                          .WORD   0
1109 003332  000000                          .WORD   0
1110 003334  000200                TAB7:     .WORD   200
1111 003336  000000                          .WORD   0
1112 003340  000000                          .WORD   0
1113 003342  000000                          .WORD   0
1114 003344  000200                TAB8:     .WORD   200
1115 003346  000000                          .WORD   0
1116 003350  000000                          .WORD   0
1117 003352  000001                          .WORD   1
1118 003354  000400  000000  000000 TAB9:     .WORD   400,0,0,0
1119 003364  030000                TAB10:    .WORD   30000
1120 003366  003000                          .WORD   3000
1121 003370  000000                          .WORD   0
1122 003372  000000                          .WORD   0
1123 003374  016400                TAB11:    .WORD   16400
1124 003376  000000                          .WORD   0
1125 003400  000000                          .WORD   0
1126 003402  000000                          .WORD   0
1127 003404  030000  003000  000002 TAB11A:   .WORD   30000,3000,2,0
1128 003414  016100  000000  000000 TAB12:    .WORD   16100,0,0,1
1129 003424  016200                TAB13:    .WORD   16200
1130 003426  000000                          .WORD   0
1131 003430  000000                          .WORD   0
1132 003432  000001                          .WORD   1
1133 003434  030000  003000  000000 TAB13B:   .WORD   30000,3000,0,140000
1134 003444  030000                TAB14:    .WORD   30000
1135 003446  000000                          .WORD   0
1136 003450  000000                          .WORD   0
1137 003452  000000                          .WORD   0
1138 003454  024700                TAB15:    .WORD   24700
1139 003456  000000                          .WORD   0
1140 003460  000000                          .WORD   0
1141 003462  000000                          .WORD   0
1142 003464  025000                TAB16:    .WORD   25000
1143 003466  175363                          .WORD   175363
1144 003470  123456                          .WORD   123456
1145 003472  123456                          .WORD   123456
1146 003474  030000                TAB17:    .WORD   30000
1147 003476  007020                          .WORD   7020
1148 003500  000000  000000                  .WORD   0,0
1149 003504  023456                TAB18:    .WORD   23456
1150 003506  000000                          .WORD   0
1151 003510  000000                          .WORD   0
1152 003512  000001                          .WORD   1
1153 003514  100200  000000  000000 TAB21:    .WORD   100200,0,0,0
1154 003524  100400  000000  000000 TAB22:    .WORD   100400,0,0,0
1155 003534  000200  000000  000000 TAB23:    .WORD   200,0,0,1
1156 003544  062400  000000  000000 TAB24:    .WORD   62400,0,0,0
1157 003554  001100  000000  000000 TAB25:    .WORD   1100,0,0,0
1158 003564  100600  000000  000000 TAB26:    .WORD   100600,0,0,0
1159 003574  001000  000000  000000 TAB27:    .WORD   1000,0,0,0
1160 003604  000600  000000  000000 TAB28:    .WORD   600,0,0,0
```

```
1161 003614  010100  000000  000000  TAB29:  .WORD   10100,0,0,0
1162 003624  010100  000000  002000  TAB29A: .WORD   10100,0,2000,0
1163                                 ;
1164 003634  000500  000000  000000  TAB30:  .WORD   500,0,0,0
1165 003644  100400  000000  000000  TAB31:  .WORD   100400,0,0,0
1166 003654  016000  000000  000000  TAB32:  .WORD   16000,0,0,0
1167 003664  011600  000000  000000  TAB33:  .WORD   11600,0,0,0
1168 003674  000640  000000  000000  TAB34:  .WORD   640,0,0,0
1169 003704  077600  000000  000000  TAB40:  .WORD   77600,0,0,0
1170 003714  100200  000000  000000  TAB41:  .WORD   100200,0,0,1
1171 003724  000340  000000  000000  TAB42:  .WORD   340,0,0,0
1172 003734  000077  177777  177777  TAB43:  .WORD   77,177777,177777,177776
1173 003744  000577  177777  177777  TAB45:  .WORD   577,-1,-1,-1
1174 003754  000577  177777  000000  TAB46:  .WORD   577,-1,0,0
1175 003764  173737  124242  052525  TAB47:  .WORD   173737,124242,052525,12346
1176 003774  000000  000000  052525  TAB47A: .WORD   0,0,052525,12346
1177 004004  173737  124242  000000  TAB48:  .WORD   173737,124242,0,0
1178 004014  000600  000000  000000  TAB49:  .WORD   600,0,0,0
1179
1180 004024                          START:
                                     ;; LCP/ORION ROUTINE TO SAVE EMTULATOR AND PRIORITY
     004024  005737  004112          EMTSAV: TST     SAV30           ;; FIRST TIME THROUGH ?
     004030  001034                          BNE     VMKOR           ;; BRANCH IF BEEN HERE ALREADY
     004032  032737  000040  000052          BIT     #BIT5,@#52      ;; ARE WE IN UFD MODE ?
     004040  001430                          BEQ     VMKOR           ;; LEAVE IF NOT
     004042  012737  177777  004116          MOV     #-1,UFDFLG      ;; SET UFD FLAG
     004050  032737  000100  000052          BIT     #BIT6,@#52      ;; ARE WE IN QUIET MODE ?
     004056  001403                          BEQ     1$              ;; BR IF NOT
     004060  012737  177777  004120          MOV     #-1,UQUIET      ;; SET QUIET MODE
     004066  104042                  1$:     EMT     42              ;; GET ADDRESS OF XXDP DCA TABLE
     004070  005060  000042                  CLR     42(R0)          ;; CLR XXDP+ "DRSERR"
     004074  013737  000030  004112          MOV     30,SAV30        ;; SAVE EMULATOR ADDRESS
     004102  013737  000032  004114          MOV     32,SAV32        ;; SAVE EMULATOR PRIORITY LEVEL
     004110  000404                          BR      VMKOR           ;; GET AROUND TAG AREA
     004112  000000                  SAV30:  .WORD   0               ;; PUT EMULATOR INFO HERE
     004114  000000                  SAV32:  .WORD   0               ;; PUT PRIORITY LOCATION        HERE
     004116  000000                  UFDFLG: .WORD   0               ;; USER FRIENDLY MODE FLAG
     004120  000000                  UQUIET: .WORD   0               ;; UFD QUIET MODE FLAG
     004122                          VMKOR:
                                     ;*******************************************************************************
1181 004122                          1$:
                                     .SBTTL  INITIALIZE THE COMMON TAGS
                                     ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
     004122  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
     004126  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
     004130  022706  001140                  CMP     #SWR,R6 ;;DONE?
     004134  001374                          BNE     .-6             ;;LOOP BACK IF NO
     004136  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                     ;;INITIALIZE A FEW VECTORS
     004142  012737  140474  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
     004150  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
     004156  012737  140776  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
     004164  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
     004172  012737  143522  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
     004200  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
     004206  012737  143604  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
     004214  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
```

INITIALIZE THE COMMON TAGS

```
        004222  013737  140406  140400          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
        004230  005037  001164                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
        004234  005037  001166                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
        004240  112737  000001  001115          MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
        004246  012737  004246  001106          MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
        004254  012737  004254  001110          MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                              ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                              ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
        004262  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
        004266  012737  004322  000004          MOV     #30000$,@#ERRVEC        ;;SET UP ERROR VECTOR
        004274  012737  177570  001140          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
        004302  012737  177570  001142          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
        004310  022777  177777  174622          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
        004316  001012                          BNE     30002$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                        ;;AND  THE HARDWARE SWR IS NOT = -1
        004320  000403                          BR      30001$          ;;BRANCH IF NO TIMEOUT
        004322  012716  004330          30000$: MOV     #30001$,(SP)    ;;SET UP FOR TRAP RETURN
        004326  000002                          RTI
        004330  012737  000176  001140  30001$: MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
        004336  012737  000174  001142          MOV     #DISPREG,DISPLAY
        004344  012637  000004          30002$: MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
        004350  005037  001206                  CLR     $PASS           ;;CLEAR PASS COUNT
        004354  132737  000200  001221          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
        004362  001403                          BEQ     30003$          ;;YES,USE NON-APT SWITCH
        004364  012737  001222  001140          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
        004372                          30003$:
1182    004372  013737  004116  004120          MOV     UFDFLG,UQUIET   ;ABORT IN UFD ON ERROR
1183    004400  012737  137542  000004          MOV     #TOUT,@#ERRVEC  ;POINT TO TIMEOUT ROUTINE
1184    004406  012737  000340  000006          MOV     #340,@#ERRVEC+2 ;AT PRIORITY 7
1185    004414  012737  137012  000114          MOV     #RAMPAR,@#114   ;POINT PARITY ABORT
1186    004422  012737  000340  000116          MOV     #340,@#116      ;AT PRIORITY7
1187    004430  012737  137724  000250          MOV     #MMUTRP,@#250   ;POINT MMU TRAP VECTOR
1188    004436  012737  000340  000252          MOV     #340,@#252      ;
1189    004444  005037  177766                  CLR     @#177766        ;CLEAR CPU ERROR REGISTER
1190    004450  032737  000100  000052          BIT     #BIT06,@#52     ;IN UFD QUIET MODE ?
1191    004456  001130                          BNE     LOOP            ;IF SO, SKIP PRINTOUT
1192    004460  012701  004472                  MOV     #.+12,R1        ;STORE LOCATION POINTER
1193    004464  012711  177777                  MOV     #-1,(R1)        ;MAKE SURE TO PRINT NOT ONLY AT FIRST TIME
1194                                    .SBTTL  TYPE PROGRAM NAME
                                        ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
        004470  005227  177777                  INC     #-1             ;;FIRST TIME?
        004474  001052                          BNE     30004$          ;;BRANCH IF NO
        004476  022737  140440  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
        004504  001446                          BEQ     30004$          ;;BRANCH IF YES
        004506  104401  004554                  TYPE    ,30005$         ;;TYPE ASCIZ STRING
                                        .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
        004512  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
        004516  001012                          BNE     30006$          ;;BRANCH IF YES
        004520  123727  001220  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
        004526  001406                          BEQ     30006$          ;;BRANCH IF YES
        004530  023727  001140  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
        004536  001005                          BNE     30007$          ;;BRANCH IF NO
        004540  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
        004542  000403                          BR      30007$
        004544  112737  000001  001134  30006$: MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
        004552                          30007$:
        004552  000423                          BR      30004$          ;;GET OVER THE ASCIZ
```

GET VALUE FOR SOFTWARE SWITCH REGISTER

```
                                        ;;30005$:       .ASCIZ  <CRLF>* KDJ11-B CPU DIAGNOSTIC - COKDADO *<CRLF>
     004622                             30004$:
1195
1196 004622  000240                            NOP                     ; debug aid
1197 004624  123727  001220  000001            CMPB    $ENV,#1         ; running under APT?
1198 004632  001020                            BNE     20$             ; default no-APT initialization
1199 004634  013700  001224                    MOV     $USWR,R0        ; work copy pass calculation
1200 004640  005700                            TST     R0              ; if = 0 default value
1201 004642  001420                            BEQ     25$             ; setup default value
1202
1203 004644  042700  000017                    BIC     #17,R0          ; clear low order nibble
1204 004650  000241                            CLC                     ; assure no unknowns
1205 004652  006000                            ROR     R0              ; 4 rotates = divide
1206 004654  006000                            ROR     R0              ; by 16 (=pass time)
1207 004656  006000                            ROR     R0              ; this area subroutined
1208 004660  006000                            ROR     R0              ; with general purpose
1209 004662  005700                            TST     R0              ; divide, this test to
1210 004664  001413                            BEQ     30$             ; determine skip altogether
1211
1212 004666  010037  003032                    MOV     R0,CCHPAS       ;residue = no. of desired passes
1213 004672  000413                            BR      35$             ; continue on
1214 004674  012737  177777  003032  20$:      MOV     #-1,CCHPAS      ; largest number no apt mode??
1215 004702  000407                            BR      35$
1216 004704  012737  000001  003032  25$:      MOV     #1,CCHPAS       ; normal default = 1
1217 004712  000403                            BR      35$
1218 004714  012737  000000  003032  30$:      MOV     #0,CCHPAS       ; no cache tests included
1219 004722  000240                  35$:      nop                     ; debug aid
1220
1221 004724  032737  000200  000052            BIT     #BIT07,@#52     ;UFD MODE?
1222 004732  001002                            BNE     LOOP            ;IF YES, BRANCH
1223 004734  004737  137070                    JSR     PC,Q22SIZ       ;SIZE FOR Q22BE
1224 004740                            LOOP:
1225                                   .DSABLE AMA
1227                                   ;;*****************************************************************
     004740  000004                    TST1:   SCOPE
1228                                   .SBTTL  BASE INSTRUCTION SET TESTS
1229                                   ;*************************************************************************
1230                                   ;*************************************************************************
1231                                   ;                       BEGIN BASE INSTRUCTION SET TESTING
1232                                   ;*************************************************************************
1233                                   ;*************************************************************************
1242 004742                            FRSTST:
1243
1244                                   ;TEST BEQ BNE INSTRUCTIONS
1245                                   ;;*****************************************************************
                                       ;THESE TWO INSTRUCTIONS ARE FUNDAMENTAL TO RECOGNIZING ERROR CONDITIONS
1246 004742  000277                            SCC
1247 004744  000244                            CLZ                             ;CC=0100 - Z BIT CLEARED
1248 004746  001401                            BEQ     1$                      ;*TEST INSTR -TRY TO CAUSE A BEQ ERROR
1249 004750  001001                            BNE     2$                      ;BRANCH IF GOOD
1250                                                                           ;THE Z FLAG DIDNT CLEAR OR BRANCH FAILED.
1251                                                                           ;FAILURE AT THIS LOCATION
1252                                                                           ;COULD MEAN A BUS PROBLEM, MICRO-CODE PROBLEM
1253                                                                           ;CONDITION CODE PROBLEM OR JUST ABOUT ANYTHING
1254                                                                           ;ELSE.
1255 004752  104001                    1$:     ERROR   +1                      ;CPU ERROR
1256 004754  000257                    2$:     CCC                             ;
```

BASE INSTRUCTION SET TESTS

```
                                        SEZ                     ;COND CODES = 0100 (ZERO)
                                        BNE     3$              ;*TEST INSTR* TRY TO BRANCH ON ZERO FLAG
                                        BEQ     4$              ;*TEST INSTR* BRANCH IF GOOD
                                                                ;BRANCH FAILURE WITH Z BIT SET
                                3$:     ERROR   +1              ;CPU ERROR
                                4$:                             ;END OF TEST
                                M2:

                                ;       TEST BRANCH ON CARRY
                                ;;*****************************************************************
                                ;THIS IS A TEST TO SEE IF THE MODULE FORM ANTICIPATED IS FEASIBLE.
                                        CCC                     ;CC=0000
                                        BCC     2$              ;*TEST INSTR*
                                                                ;BRANCH CARRY CLEAR FAILED
                                1$:     ERROR   +1              ;CPU ERROR
                                2$:     SEC                     ;CC=1111
                                        BCS     4$              ;*TEST INSTR*
                                                                ; BRANCH CARRY SET FAILED
                                3$:     ERROR   +1              ;CPU ERROR
                                4$:

                                M3:

                                ;       TEST DATA PATHS
                                        CLR     R0
                                        BNE     1$              ;TRY TO INSURE WE ARE TESTING
                                                                ;THE DATA PATH AND NOT THE "CLR R0" INSTRUCTION
                                        CLR     (R0)            ;FORCE LOCATION TO ZERO
                                        BNE     1$              ;TRY TO INSURE 0=0
                                        TST     @#0             ;AGAIN, TRY TO INSURE THAT 0=0
                                        BEQ     2$              ;BRANCH IF GOOD
                                                                ;LOCATION 0 NOT SETUP PROPERLY
                                1$:     ERROR   +1              ;CPU ERROR
                                2$:
                                M4:

                                ;       TEST DATA PATHS - ONES AND ZEROS
                                        MOV     #125252,@#0             ;0=125252
                                        CMP     #125252,@#0             ;SEE IF DATA MADE IT
                                        BEQ     2$                      ;BRANCH IF IF DATA IS GOOD
                                                                        ;ERROR! EITHER THE BUS IS BAD,
                                                                        ;OR THE MOV OR COMPARE
                                                                        ;INSTRUCTIONS FAILED
                                1$:     ERROR   +1              ;CPU ERROR
                                2$:                             ;END OF TEST

                                ;
                                M5:

                                ;       TEST DATA PATHS - DATA 0'S AND 1'S
                                        MOV     #052525,@#0             ;SETUP DATA
                                        CMP     @#0,#052525             ; TEST FOR CORRECT DATA
                                        BEQ     2$
                                1$:     ERROR   +1              ;CPU ERROR
                                2$:
                                ;
                                M6:
```

BASE INSTRUCTION SET TESTS

```
                                        ;       TEST DATA PATHS - 1'S
005062  005037  000000                        CLR     @#0
005066  005137  000000                        COM     @#0                     ;SET UP MEMORY LOCATION 0 = 111111
005072  023727  000000  177777                CMP     @#0,#177777             ; TEST DATA
005100  001401                                BEQ     2$                      ;BRANCH IF NO ERROR
005102  104001                        1$:     ERROR   +1                      ;CPU ERROR
005104                                2$:
                                        ;
005104                                GPROTS:
                                        ;       R0 BIT TESTS
005104  012700  177777                        MOV     #177777,R0                      ;R0=177777
005110  020027  177777                        CMP     R0,#177777                      ;DOES R0=177777
005114  001401                                BEQ     1$                              ;YES GO ON
                                                                                      ;NO GO TO ERROR
005116  104001                                ERROR   +1                      ;CPU ERROR
005120  005000                        1$:     CLR     R0                              ;R0=0
005122  020027  000000                        CMP     R0,#0                           ;DOES R0=0
005126  001401                                BEQ     2$                              ;YES GO ON
                                                                                      ;NO GO TO ERROR
005130  104001                                ERROR   +1                      ;CPU ERROR
005132  012700  125252                2$:     MOV     #125252,R0                      ;R0=125252
005136  020027  125252                        CMP     R0,#125252                      ;DOES R0=125252
005142  001401                                BEQ     3$                              ;YES GO ON
                                                                                      ;NO GO TO ERROR
005144  104001                                ERROR   +1                      ;CPU ERROR
005146  012700  052525                3$:     MOV     #52525,R0                       ;R0=52525
005152  020027  052525                        CMP     R0,#52525                       ;DOES R0=52525
005156  001401                                BEQ     4$                              ;YES GO ON
                                                                                      ;NO GO TO ERROR
005160  104001                                ERROR   +1                      ;CPU ERROR
005162                                4$:
                                        ;
005162                                GPR1TS:
                                        ;       R1 BIT TESTS
005162  012701  177777                        MOV     #177777,R1                      ;R1=177777
005166  020127  177777                        CMP     R1,#177777                      ;DOES R1=177777
005172  001401                                BEQ     1$                              ;YES GO ON
                                                                                      ;NO GO TO ERROR
005174  104001                                ERROR   +1                      ;CPU ERROR
005176  005001                        1$:     CLR     R1                              ;R1=0
005200  020127  000000                        CMP     R1,#0                           ;DOES R1=0
005204  001401                                BEQ     2$                              ;YES GO ON
                                                                                      ;NO GO TO ERROR
005206  104001                                ERROR   +1                      ;CPU ERROR
005210  012701  125252                2$:     MOV     #125252,R1                      ;R1=125252
005214  020127  125252                        CMP     R1,#125252                      ;DOES R1=125252
005220  001401                                BEQ     3$                              ;YES GO ON
                                                                                      ;NO GO TO ERROR
005222  104001                                ERROR   +1                      ;CPU ERROR
005224  012701  052525                3$:     MOV     #52525,R1                       ;R1=52525
005230  020127  052525                        CMP     R1,#52525                       ;DOES R1=52525
005234  001401                                BEQ     4$                              ;YES GO ON
                                                                                      ;NO GO TO ERROR
005236  104001                                ERROR   +1                      ;CPU ERROR
005240                                4$:
                                        ;
```

BASE INSTRUCTION SET TESTS

```
1393 005240                            GPR2TS:
1394                                   ;       R2 BIT TESTS
1395 005240  012702  177777                    MOV     #177777,R2              ;R2=177777
1396 005244  020227  177777                    CMP     R2,#177777              ;DOES R2=177777
1397 005250  001401                            BEQ     1$                      ;YES GO ON
1398                                                                           ;NO GO TO ERROR
1399 005252  104001                            ERROR   +1                  ;CPU ERROR
1400 005254  005002                    1$:     CLR     R2                      ;R2=0
1401 005256  020227  000000                    CMP     R2,#0                   ;DOES R2=0
1402 005262  001401                            BEQ     2$                      ;YES GO ON
1403                                                                           ;NO GO TO ERROR
1404 005264  104001                            ERROR   +1                  ;CPU ERROR
1405 005266  012702  125252            2$:     MOV     #125252,R2              ;R2=125252
1406 005272  020227  125252                    CMP     R2,#125252              ;DOES R2=125252
1407 005276  001401                            BEQ     3$                      ;YES GO ON
1408                                                                           ;NO GO TO ERROR
1409 005300  104001                            ERROR   +1                  ;CPU ERROR
1410 005302  012702  052525            3$:     MOV     #52525,R2               ;R2=52525
1411 005306  020227  052525                    CMP     R2,#52525               ;DOES R2=52525
1412 005312  001401                            BEQ     4$                      ;YES GO ON
1413                                                                           ;NO GO TO ERROR
1414 005314  104001                            ERROR   +1                  ;CPU ERROR
1415 005316                            4$:
1416                                   ;
1419 005316                            GPR3TS:
1420                                   ;       R3 BIT TESTS
1421 005316  012703  177777                    MOV     #177777,R3              ;R3=177777
1422 005322  020327  177777                    CMP     R3,#177777              ;DOES R3=177777
1423 005326  001401                            BEQ     1$                      ;YES GO ON
1424                                                                           ;NO GO TO ERROR
1425 005330  104001                            ERROR   +1                  ;CPU ERROR
1426 005332  005003                    1$:     CLR     R3                      ;R3=0
1427 005334  020327  000000                    CMP     R3,#0                   ;DOES R3=0
1428 005340  001401                            BEQ     2$                      ;YES GO ON
1429                                                                           ;NO GO TO ERROR
1430 005342  104001                            ERROR   +1                  ;CPU ERROR
1431 005344  012703  125252            2$:     MOV     #125252,R3              ;R3=125252
1432 005350  020327  125252                    CMP     R3,#125252              ;DOES R3=125252
1433 005354  001401                            BEQ     3$                      ;YES GO ON
1434                                                                           ;NO GO TO ERROR
1435 005356  104001                            ERROR   +1                  ;CPU ERROR
1436 005360  012703  052525            3$:     MOV     #52525,R3               ;R3=52525
1437 005364  020327  052525                    CMP     R3,#52525               ;DOES R3=52525
1438 005370  001401                            BEQ     4$                      ;YES GO ON
1439                                                                           ;NO GO TO ERROR
1440 005372  104001                            ERROR   +1                  ;CPU ERROR
1441 005374                            4$:
1442                                   ;
1445 005374                            GPR4TS:
1446                                   ;       R4 BIT TESTS
1447 005374  012704  177777                    MOV     #177777,R4              ;R4=177777
1448 005400  020427  177777                    CMP     R4,#177777              ;DOES R4=177777
1449 005404  001401                            BEQ     1$                      ;YES GO ON
1450                                                                           ;NO GO TO ERROR
1451 005406  104001                            ERROR   +1                  ;CPU ERROR
1452 005410  005004                    1$:     CLR     R4                      ;R4=0
1453 005412  020427  000000                    CMP     R4,#0                   ;DOES R4=0
```

BASE INSTRUCTION SET TESTS

```
1454 005416  001401                    BEQ     2$                      ;YES GO ON
1455                                                                   ;NO GO TO ERROR
1456 005420  104001                    ERROR   +1              ;CPU ERROR
1457 005422  012704  125252      2$:   MOV     #125252,R4              ;R4=125252
1458 005426  020427  125252            CMP     R4,#125252              ;DOES R4=125252
1459 005432  001401                    BEQ     3$                      ;YES GO ON
1460                                                                   ;NO GO TO ERROR
1461 005434  104001                    ERROR   +1              ;CPU ERROR
1462 005436  012704  052525      3$:   MOV     #52525,R4               ;R4=52525
1463 005442  020427  052525            CMP     R4,#52525               ;DOES R4=52525
1464 005446  001401                    BEQ     4$                      ;YES GO ON
1465                                                                   ;NO GO TO ERROR
1466 005450  104001                    ERROR   +1              ;CPU ERROR
1467 005452                      4$:
1468                                    ;
1471 005452                      GPR5TS:
1472                                    ;       R5 BIT TESTS
1473 005452  012705  177777            MOV     #177777,R5              ;R5=177777
1474 005456  020527  177777            CMP     R5,#177777              ;DOES R5=177777
1475 005462  001401                    BEQ     1$                      ;YES GO ON
1476                                                                   ;NO GO TO ERROR
1477 005464  104001                    ERROR   +1              ;CPU ERROR
1478 005466  005005              1$:   CLR     R5                      ;R5=0
1479 005470  020527  000000            CMP     R5,#0                   ;DOES R5=0
1480 005474  001401                    BEQ     2$                      ;YES GO ON
1481                                                                   ;NO GO TO ERROR
1482 005476  104001                    ERROR   +1              ;CPU ERROR
1483 005500  012705  125252      2$:   MOV     #125252,R5              ;R5=125252
1484 005504  020527  125252            CMP     R5,#125252              ;DOES R5=125252
1485 005510  001401                    BEQ     3$                      ;YES GO ON
1486                                                                   ;NO GO TO ERROR
1487 005512  104001                    ERROR   +1              ;CPU ERROR
1488 005514  012705  052525      3$:   MOV     #52525,R5               ;R5=52525
1489 005520  020527  052525            CMP     R5,#52525               ;DOES R5=52525
1490 005524  001401                    BEQ     4$                      ;YES GO ON
1491                                                                   ;NO GO TO ERROR
1492 005526  104001                    ERROR   +1              ;CPU ERROR
1493 005530                      4$:
1494                                    ;
1497 005530                      GPR6TS:
1498                                    ;       R6 BIT TESTS
1499 005530  012706  177777            MOV     #177777,R6              ;R6=177777
1500 005534  020627  177777            CMP     R6,#177777              ;DOES R6=177777
1501 005540  001401                    BEQ     1$                      ;YES GO ON
1502                                                                   ;NO GO TO ERROR
1503 005542  104001                    ERROR   +1              ;CPU ERROR
1504 005544  005006              1$:   CLR     R6                      ;R6=0
1505 005546  020627  000000            CMP     R6,#0                   ;DOES R6=0
1506 005552  001401                    BEQ     2$                      ;YES GO ON
1507                                                                   ;NO GO TO ERROR
1508 005554  104001                    ERROR   +1              ;CPU ERROR
1509 005556  012706  125252      2$:   MOV     #125252,R6              ;R6=125252
1510 005562  020627  125252            CMP     R6,#125252              ;DOES R6=125252
1511 005566  001401                    BEQ     3$                      ;YES GO ON
1512                                                                   ;NO GO TO ERROR
1513 005570  104001                    ERROR   +1              ;CPU ERROR
1514 005572  012706  052525      3$:   MOV     #52525,R6               ;R6=52525
```

```
 1515 005576  020627  052525                    CMP     R6,#52525               ;DOES R6=52525
 1516 005602  001401                            BEQ     4$                      ;YES GO ON
 1517                                                                           ;NO GO TO ERROR
 1518 005604  104001                            ERROR   .+1             ;CPU ERROR
 1519 005606  012706  001000          4$:       MOV     #STBOT,R6               ;RESTORE SP
 1520
 1521                                 ;
 1524 005612                          PSWBTS:
 1525                                 ;       PSW LOW BYTE BIT TESTS
 1526 005612  012737  000377  177776            MOV     #377,@#177776           ;PS=357 T BIT SHOULDN'T SET
 1527 005620  022737  000357  177776            CMP     #357,@#177776           ;DOES PS=357
 1528 005626  001401                            BEQ     1$                      ;YES GO ON
 1529                                                                           ;NO GO TO ERROR
 1530 005630  104001                            ERROR   .+1             ;CPU ERROR
 1531 005632  005037  177776          1$:       CLR     @#177776                ;PS=0
 1532 005636  022737  000000  177776            CMP     #0,@#177776             ;DOES PS=0
 1533 005644  001401                            BEQ     2$                      ;YES GO ON
 1534                                                                           ;NO GO TO ERROR
 1535 005646  104001                            ERROR   .+1             ;CPU ERROR
 1536 005650  012737  000105  177776  2$:       MOV     #105,@#177776           ;PS=105
 1537 005656  022737  000105  177776            CMP     #105,@#177776           ;DOES PS=105
 1538 005664  001401                            BEQ     3$                      ;YES GO ON
 1539                                                                           ;NO GO TO ERROR
 1540 005666  104001                            ERROR   .+1             ;CPU ERROR
 1541 005670  012737  000252  177776  3$:       MOV     #252,@#177776           ;PS=252
 1542 005676  022737  000252  177776            CMP     #252,@#177776           ;DOES PS=252
 1543 005704  001401                            BEQ     4$                      ;YES GO ON
 1544                                                                           ;NO GO TO ERROR
 1545 005706  104001                            ERROR   .+1             ;CPU ERROR
 1546 005710                          4$:
 1556
 1557 005710                          MSP0:
 1558
 1559                                 ;       TEST SINGLE OPERAND INSTRUCTIONS- MODE 0
 1560                                 ;;*****************************************************************
                                      ;THE INC, COM, CLR, AND DECREMENT INSTRUCTIONS ARE VERIFED.
 1561 005710  005004                            CLR     R4                              ;INITIALIZE R4 WITH DATA
 1562 005712  005104                            COM     R4                              ;
 1563 005714  005004                            CLR     R4                              ;*TEST INSTRUCTION
 1564 005716  001401                            BEQ     2$                              ;BRANCH IF R4 CLEARED
 1565 005720  104001                  1$:       ERROR   .+1             ;CPU ERROR
 1566 005722  005104                  2$:       COM     R4                              ;*TEST COMPLIMENT INSTRUCTION
 1567 005724  005204                            INC     R4                              ;*TEST INCREMENT INSTRUCTION
 1568 005726  001401                            BEQ     4$                              ;BRANCH IF R4 =0
 1569                                                                                   ;COMPLIMENT OR INCREMENT FAILED
 1570 005730  104001                  3$:       ERROR   .+1             ;CPU ERROR
 1571 005732                          4$:
 1572
 1573                                 ;
 1576 005732                          MSPB:
 1577
 1578                                 ;       TEST SINGLE OPS - EVEN BYTE OF CLRB, DECB, AND COMB
 1579 005732  005004                            CLR     R4
 1580 005734  005104                            COM     R4                      ;SETUP TEST REGISTER
 1581 005736  105004                            CLRB    R4                      ;*TEST CLEAR BYTE INSTRUCTION
 1582 005740  001401                            BEQ     2$                      ;BRANCH IF GOOD
 1583                                                                           ;CLEAR EVEN BYTE FAILED
```

```
1584 005742  104001                1$:     ERROR   +1              ;CPU ERROR
1585 005744  105304                2$:     DECB    R4              ;*TEST DECREMENT BYTE
1586 005746  100002                        BPL     3$              ;DECREMENT BYTE FAILED
1587 005750  105104                        COMB    R4              ;*TEST COMPLIMENT BYTE
1588 005752  001401                        BEQ     4$              ;BRANCH IF GOOD
1589                                                               ;COMPLIMENT OR DECREMENT FAILED TO WORK
1590 005754  104001                3$:     ERROR   +1              ;CPU ERROR
1591 005756                        4$:
1592
1593                               ;
1596 005756                        MSPC:
1597                               ;       TEST SINGLE OPS - MODE 1 CLRB, COMB, AND INCB
1598 005756  005004                        CLR     R4
1599 005760  005014                        CLR     (R4)
1600 005762  005114                        COM     (R4)            ;SETUP TEST DATA
1601 005764  005014                        CLR     (R4)            ;*TEST INSTRUCTION
1602 005766  001401                        BEQ     2$              ;BRANCH IF GOOD
1603                                                               ;MODE 1 FAILED
1604 005770  104001                1$:     ERROR   +1              ;CPU ERROR
1605 005772  005114                2$:     COM     (R4)            ;*TEST INSTRUCTION
1606 005774  001403                        BEQ     3$              ;(0)SHOULD = -1
1607 005776  100002                        BPL     3$              ;
1608 006000  005214                        INC     (R4)            ;*TEST INSTRUCTION
1609 006002  001401                        BEQ     4$              ;BRANCH IF GOOD
1610                                                               ;COM OR INC FAILED TO ALTER LOC 0 CORRECTLY
1611 006004  104001                3$:     ERROR   +1              ;CPU ERROR
1612 006006                        4$:
1613
1614                               ;
1617 006006                        MSPD:
1618
1619                               ;       TEST SINGLE OPS MODE1-EVEN BYTE-CLRB,COMB,INCB
1620 006006  005004                        CLR     R4
1621 006010  005014                        CLR     (R4)
1622 006012  005114                        COM     (R4)            ;SETUP TEST DATA
1623 006014  105014                        CLRB    (R4)            ;*TEST INSTRUCTION
1624 006016  105014                        CLRB    (R4)            ;*TEST INSTRUCTION
1625 006020  001401                        BEQ     2$              ;BRANCH IF GOOD
1626                                                               ;CLEAR (0) EVEN BYTE FAILED
1627 006022  104001                1$:     ERROR   +1              ;CPU ERROR
1628 006024  105214                2$:     INCB    (R4)            ;*TEST INSTRUCTION
1629 006026  100405                        BMI     3$              ; TEST FLAGS
1630 006030  001404                        BEQ     3$
1631 006032  105114                        COMB    (R4)            ;*TEST INSTRUCTION
1632 006034  105214                        INCB    (R4)
1633 006036  105214                        INCB    (R4)
1634 006040  001401                        BEQ     4$              ;BRANCH IF GOOD
1635                                                               ;COMB OR INCB FAILED
1636 006042  104001                3$:     ERROR   +1              ;CPU ERROR
1637 006044                        4$:
1638
1639                               ;
1642 006044                        MSPEO:
1643
1644                               ;       TEST SINGLE OPS - ODD BYTE - CLRB, COMB, DECB
1645 006044  005004                        CLR     R4
1646 006046  005014                        CLR     (R4)
```

BASE INSTRUCTION SET TESTS

```
1647 006050  005114                    COM     (R4)            ;SETUP TEST DATA
1648 006052  005204                    INC     R4              ;POINT TO ODD BYTE
1649 006054  105014                    CLRB    (R4)            ;*TEST INSTRUCTION
1650 006056  001401                    BEQ     1$              ;BRANCH IF GOOD
1651                                                           ;CLEAR ODD BYTE FAILED
1652 006060  104001                    ERROR   +1              ;CPU ERROR
1653 006062  005304            1$:     DEC     R4              ;POINT TO EVEN BYTE
1654 006064  005214                    INC     (R4)            ;LOC 0=1 0
1655 006066  005204                    INC     R4              ;POINT TO ODD BYTE
1656 006070  105114                    COMB    (R4)            ;*TEST INSTRUCTION
1657 006072  105214                    INCB    (R4)            ;LOC 0=-1 0
1658 006074  100003                    BPL     2$              ;BRANCH IF ERROR
1659 006076  001402                    BEQ     2$              ;
1660 006100  105214                    INCB    (R4)            ;*TEST INSTRUCTION
1661 006102  001401                    BEQ     3$              ;BRANCH IF GOOD
1662                                                           ;MODE 1, ODD BYTE FAILED
1663 006104  104001            2$:     ERROR   +1              ;CPU ERROR
1664 006106                    3$:
1665
1666                           ;
1669 006106                    MSPF:
1670                           ;       TEST SINGLE OP - MODE 2 - CLR, COM, INC
1671 006106  005004                    CLR     R4
1672 006110  105104                    COMB    R4
1673 006112  005204                    INC     R4              ;R4=400
1674 006114  005014                    CLR     (R4)            ;400=0
1675 006116  005114                    COM     (R4)            ;400=-1
1676 006120  005024                    CLR     (R4)+           ;*TEST INSTRUCTION
1677 006122  001401                    BEQ     1$              ;BRANCH IF GOOD
1678                                                           ;MODE 2 CLEAR FAILED
1679 006124  104001                    ERROR   +1              ;CPU ERROR
1680 006126  005304            1$:     DEC     R4
1681 006130  005304                    DEC     R4              ;R4=400
1682 006132  005124                    COM     (R4)+           ;*TEST INSTRUCTION
1683 006134  100004                    BPL     2$              ;BRANCH IF FAILURE
1684 006136  005304                    DEC     R4
1685 006140  005304                    DEC     R4              ;R4=400
1686 006142  005224                    INC     (R4)+           ;*TEST INSTRUCTION
1687 006144  001401                    BEQ     3$              ;BRANCH IF GOOD
1688                                                           ;MODE 2 FAILURE
1689 006146  104001            2$:     ERROR   +1              ;CPU ERROR
1690 006150                    3$:
1691
1692                           ;
1695 006150                    MSPG:
1696
1697                           ;       TEST CLRB, COMB, DECB, MODE 2 - EVEN BYTE
1698 006150  005004                    CLR     R4
1699 006152  105104                    COMB    R4
1700 006154  005204                    INC     R4              ;R4=400
1701 006156  005014                    CLR     (R4)
1702 006160  005114                    COM     (R4)            ;400=-1
1703 006162  105024                    CLRB    (R4)+           ;*TEST INSTRUCTION
1704 006164  001401                    BEQ     1$              ;BRANCH IF GOOD
1705                                                           ;MODE 2 EVEN BYTE FAILED
1706 006166  104001                    ERROR   +1              ;CPU ERROR
1707 006170  005304            1$:     DEC     R4
```

BASE INSTRUCTION SET TESTS

```
1708 006172  105324                    DECB    (R4)+           ;*TEST INSTRUCTION
1709 006174  100003                    BPL     2$              ;BRANCH IF BAD
1710 006176  005304                    DEC     R4              ;POINT TO EVEN BYTE
1711 006200  105124                    COMB    (R4)+           ;*TEST INSTRUCTION
1712 006202  001401                    BEQ     3$              ;BRANCH IF GOOD
1713                                                           ;MODE 2, EVEN BYTE FAILED
1714 006204  104001            2$:     ERROR   +1              ;CPU ERROR
1715 006206                    3$:
1716
1717                           ;
1720 006206                    MSPH:
1721
1722                           ;       TEST CLRB, COMB, INCB MODE 2 - ODD BYTE
1723 006206  005004                    CLR     R4
1724 006210  105104                    COMB    R4
1725 006212  005204                    INC     R4              ;R4=400
1726 006214  005014                    CLR     (R4)
1727 006216  005114                    COM     (R4)            ;400=-1 -1
1728 006220  005214                    INC     (R4)            ;POINT TO ODD BYTE
1729 006222  105024                    CLRB    (R4)+           ;*TEST INSTRUCTION
1730 006224  001401                    BEQ     1$              ;BRANCH IF GOOD
1731                                                           ;MODE 2,ODD BYTE FAILED
1732 006226  104001                    ERROR   +1              ;CPU ERROR
1733 006230  005304            1$:     DEC     R4              ;POINT TO ODD BYTE
1734 006232  005304                    DEC     R4
1735 006234  105224                    INCB    (R4)+           ;400=1 0
1736 006236  105124                    COMB    (R4)+           ;*TEST INSTRUCTION
1737 006240  100003                    BPL     2$              ;BRANCH IF MODE 2 FAILED
1738 006242  005304                    DEC     R4              ;POINT TO ODD BYTE
1739 006244  105224                    INCB    (R4)+
1740 006246  001401                    BEQ     3$              ;BRANCH IF GOOD
1741                                                           ;MODE 2,ODD BYTE FAILED
1742 006250  104001            2$:     ERROR   +1              ;CPU ERROR
1743 006252                    3$:
1744
1745                           ;
1748 006252                    MSPI:
1749
1750                           ;       TEST CLR, COM, INC - MODE 3
1751 006252  005004                    CLR     R4              ;
1752 006254  005014                    CLR     (R4)            ;0=0
1753 006256  105114                    COMB    (R4)            ;
1754 006260  005214                    INC     (R4)            ;0=400
1755 006262  005034                    CLR     @(R4)+          ;*TEST INSTRUCTION
1756 006264  001401                    BEQ     1$              ;BRANCH IF GOOD
1757                                                           ;MODE 3 FAILED, 400 SHOULD=0
1758 006266  104001                    ERROR   +1              ;CPU ERROR
1759 006270  005304            1$:     DEC     R4
1760 006272  005304                    DEC     R4              ;R4=0
1761 006274  005134                    COM     @(R4)+          ;*TEST INSTRUCTION
1762 006276  100004                    BPL     2$              ;BRANCH IF BAD
1763 006300  005304                    DEC     R4
1764 006302  005304                    DEC     R4              ;REPOSITION POINTER
1765 006304  005234                    INC     @(R4)+          ;*TEST INSTRUCTION
1766 006306  001401                    BEQ     3$              ;BRANCH IF GOOD
1767                                                           ;MODE 3 FAILED
1768 006310  104001            2$:     ERROR   +1              ;CPU ERROR
```

```
1769 006312                        3$:
1770
1771                               ;
1774 006312                        MSPJ:
1775
1776                               ;       TEST CLRB, COMB, INCB - MODE 3, EVEN/ODD BYTE
1777 006312  005004                        CLR     R4              ;R4=0
1778 006314  005001                        CLR     R1
1779 006316  105101                        COMB    R1
1780 006320  005201                        INC     R1              ;R1=400
1781 006322  005011                        CLR     (R1)
1782 006324  005121                        COM     (R1)+           ;400=-1
1783 006326  005011                        CLR     (R1)            ;
1784 006330  105111                        COMB    (R1)            ;402=000 377
1785 006332  005014                        CLR     (R4)            ;
1786 006334  105114                        COMB    (R4)            ;
1787 006336  005214                        INC     (R4)            ;0=400
1788 006340  105034                        CLRB    @(R4)+          ;*TEST INSTRUCTION 400=377 000
1789 006342  001401                        BEQ     1$              ;BRANCH IF MODE 3 EVEN BYTE CLEARED
1790                                                               ; TEST INSTRUCTION FAILED
1791 006344  104001                        ERROR   +1              ;CPU ERROR
1792 006346  005304                1$:     DEC     R4              ;REPOSTION POINTER
1793 006350  005304                        DEC     R4
1794 006352  105134                        COMB    @(R4)+          ;*TEST INSTRUCTION
1795 006354  005304                        DEC     R4
1796 006356  005304                        DEC     R4              ;REPOSITION POINTER
1797 006360  105234                        INCB    @(R4)+          ;*TEST INSTRUCTION
1798 006362  001401                        BEQ     3$              ;BRANCH IF GOOD
1799                                                               ;MODE 3, EVEN BYTE FAILED
1800 006364  104001                2$:     ERROR   +1              ;CPU ERROR
1801 006366  005304                3$:     DEC     R4
1802 006370  005304                        DEC     R4
1803 006372  005214                        INC     (R4)            ;R4=401
1804 006374  105234                        INCB    @(R4)+          ;*TEST INSTRUCTION
1805 006376  001004                        BNE     4$              ;BRANCH IF 402 NEQ 0
1806 006400  005304                        DEC     R4
1807 006402  005304                        DEC     R4              ;R4=401
1808 006404  105034                        CLRB    @(R4)+          ;401=0
1809 006406  001401                        BEQ     5$              ;BRANCH IF GOOD
1810                                                               ;ODD BYTE FAILED
1811 006410  104001                4$:     ERROR   +1              ;CPU ERROR
1812 006412  005304                5$:     DEC     R4
1813 006414  005304                        DEC     R4              ;R4=401
1814 006416  105134                        COMB    @(R4)+          ;403=377
1815 006420  005304                        DEC     R4
1816 006422  005304                        DEC     R4
1817 006424  105234                        INCB    @(R4)+          ;*TEST INSTRUCTION
1818 006426  001401                        BEQ     7$              ;BRANCH IF GOOD
1819                                                               ;MODE3 ODD BYTE FAILED.
1820 006430  104001                6$:     ERROR   +1              ;CPU ERROR
1821 006432                        7$:
1822
1823                               ;
1826 006432                        MSPL:
1827
1828                               ;       TEST CLR, COM, DEC - MODE 4
1829 006432  005004                        CLR     R4
```

BASE INSTRUCTION SET TESTS

```
1830 006434  105104                    COMB    R4
1831 006436  005204                    INC     R4              ;R4=400
1832 006440  005014                    CLR     (R4)            ;
1833 006442  005124                    COM     (R4)+           ;400=-1
1834 006444  005014                    CLR     (R4)            ;
1835 006446  005224                    INC     (R4)+           ;402=1
1836 006450  005044                    CLR     -(R4)           ;*TEST INSTRUCTION
1837 006452  001401                    BEQ     1$              ;BRANCH IF GOOD
1838                                                           ;MODE 4 FAILED
1839 006454  104001                    ERROR   +1              ;CPU ERROR
1840 006456  005344            1$:     DEC     -(R4)           ;*TEST INSTRUCTION 400=-2
1841 006460  005114                    COM     (R4)            ;400=1
1842 006462  001405                    BEQ     2$              ;BRANCH IF BAD
1843 006464  100404                    BMI     2$              ;BRANCH IF BAD
1844 006466  005204                    INC     R4
1845 006470  005204                    INC     R4              ;R4=400
1846 006472  005344                    DEC     -(R4)           ;*TEST INSTRUCTION
1847 006474  001401                    BEQ     3$              ;BRANCH IF GOOD
1848                                                           ;MODE 4 FAILED
1849 006476  104001            2$:     ERROR   +1              ;CPU ERROR
1850 006500                    3$:
1851
1852                           ;
1855 006500                    MSPM:
1856
1857                           ;       TEST COMB, INCB, CLRB - MODE 4, ODD BYTE
1858 006500  005004                    CLR     R4
1859 006502  105104                    COMB    R4              ;
1860 006504  005204                    INC     R4              ;R4=400
1861 006506  005044                    CLR     -(R4)           ;376=0
1862 006510  105114                    COMB    (R4)
1863 006512  005224                    INC     (R4)+           ;376=001 000
1864 006514  005014                    CLR     (R4)
1865 006516  005124                    COM     (R4)+           ;400=1
1866 006520  005204                    INC     R4              ;R4=403
1867 006522  105044                    CLRB    -(R4)           ; TEST INST. CLEAR ODD BYTE (401)
1868 006524  001401                    BEQ     2$              ;BRANCH IF GOOD
1869                                                           ;MODE 4 BYTE FAILED
1870 006526  104001            1$:     ERROR   +1              ;CPU ERROR
1871 006530  005204            2$:     INC     R4
1872 006532  005204                    INC     R4              ;R4=403
1873 006534  105144                    COMB    -(R4)           ; TEST INST. 401=377
1874 006536  005304                    DEC     R4
1875 006540  005304                    DEC     R4
1876 006542  105244                    INCB    -(R4)           ; TEST INST. 401=0
1877 006544  001401                    BEQ     4$              ;BRANCH IF GOOD
1878                                                           ;MODE 4 ODD BYTE FAILED
1879 006546  104001            3$:     ERROR   +1              ;CPU ERROR
1880 006550  105344            4$:     DECB    -(R4)           ;*TEST INST.
1881 006552  001401                    BEQ     6$              ;BRANCH IF GOOD
1882                                                           ;MODE 4 DECREMENT ODD BYTE FAILED
1883 006554  104001            5$:     ERROR   +1              ;CPU ERROR
1884 006556                    6$:
1885
1886
1889 006556                    MSPN:
1890
```

```
1891                                    ;       TEST CLR, COM, INC - MODE 5
1892 006556  005004                             CLR     R4
1893 006560  005014                             CLR     (R4)
1894 006562  105114                             COMB    (R4)
1895 006564  005224                             INC     (R4)+           ;0=400
1896 006566  005054                             CLR     @-(R4)          ;*TEST INST. 400=0
1897 006570  001401                             BEQ     1$              ;BRANCH IF GOOD
1898                                                                    ;MODE 5 FAILED
1899 006572  104001                             ERROR   +1              ;CPU ERROR
1900 006574  005204                     1$:     INC     R4
1901 006576  005204                             INC     R4              ;RESET POINTER TO 0
1902 006600  005154                             COM     @-(R4)          ;*TEST INST. 376=1
1903 006602  001407                             BEQ     2$              ;BRANCH IF BAD
1904 006604  005204                             INC     R4
1905 006606  005204                             INC     R4              ;REPOSITION POINTER
1906 006610  005354                             DEC     @-(R4)          ;*TEST INST. 376=0
1907 006612  001403                             BEQ     2$              ;BRANCH IF BAD
1908 006614  005224                             INC     (R4)+           ;0=401 R4=2
1909 006616  105254                             INCB    @-(R4)          ;*TEST INST.,400= 0 376
1910 006620  001401                             BEQ     3$              ;BRANCH IF GOOD
1911                                                                    ;MODE 5 FAILED
1912 006622  104001                     2$:     ERROR   +1              ;CPU ERROR
1913 006624                             3$:
1914
1915                                    ;
1918 006624                             MSPO:
1919
1920                                    ;       TEST NEG MODE 5
1921 006624  005004                             CLR     R4
1922 006626  105104                             COMB    R4
1923 006630  005204                             INC     R4              ;R4=400
1924 006632  005001                             CLR     R1
1925 006634  105101                             COMB    R1
1926 006636  005301                             DEC     R1              ;R1=376
1927 006640  005002                             CLR     R2              ;R2=0
1928 006642  005012                             CLR     (R2)            ;0=0
1929 006644  005014                             CLR     (R4)            ;
1930 006646  005114                             COM     (R4)            ;400=-1
1931 006650  005011                             CLR     (R1)            ;376=0
1932 006652  005454                             NEG     @-(R4)          ;0=0
1933 006654  001401                             BEQ     2$              ;BRANCH IF GOOD
1934                                                                    ;NEG FAILED
1935 006656  104001                     1$:     ERROR   +1              ;CPU ERROR
1936 006660  005334                     2$:     DEC     @(R4)+          ;0=-1
1937 006662  005454                             NEG     @-(R4)          ;0=1
1938 006664  001403                             BEQ     3$              ;BRANCH IF BAD
1939 006666  102402                             BVS     3$              ;BRANCH IF BAD
1940 006670  100401                             BMI     3$              ;BRANCH IF BAD
1941 006672  103401                             BCS     4$              ;BRANCH IF GOOD
1942                                                                    ;NEG FAILED
1943 006674  104001                     3$:     ERROR   +1              ;CPU ERROR
1944 006676  005334                     4$:     DEC     @(R4)+          ; TEST RESULT OF NEGATE
1945 006700  001401                             BEQ     6$              ;BRANCH IF GOOD
1946                                                                    ;RESULT OF NEGATE BAD
1947 006702  104001                     5$:     ERROR   +1              ;CPU ERROR
1948 006704  105212                     6$:     INCB    (R2)            ;0=1
1949 006706  005454                             NEG     @-(R4)          ;0=-1
```

BASE INSTRUCTION SET TESTS

```
1950 006710  001403                    BEQ     7$              ;
1951 006712  102402                    BVS     7$
1952 006714  103001                    BCC     7$              ;
1953 006716  100401                    BMI     8$              ;BRANCH IF GOOD
1954                                                           ;BAD NEGATE
1955 006720  104001            7$:     ERROR   +1              ;CPU ERROR
1956 006722  105212            8$:     INCB    (R2)            ;0=0
1957 006724  001401                    BEQ     10$             ;BRANCH IF GOOD
1958 006726  104001            9$:     ERROR   +1              ;CPU ERROR
1959 006730                    10$:
1960
1961                           ;
1963
1965 006730                    MSPP:
1966
1967                           ;       TEST CLR, COM, INC - MODE 6
1968 006730  005004                    CLR     R4
1969 006732  005204                    INC     R4
1970 006734  005204                    INC     R4              ;R4=2
1971 006736  005001                    CLR     R1
1972 006740  105101                    COMB    R1
1973 006742  005201                    INC     R1              ;R1=400
1974 006744  005011                    CLR     (R1)            ;
1975 006746  005121                    COM     (R1)+           ;400=-1
1976 006750  005011                    CLR     (R1)            ;R1=402
1977 006752  005211                    INC     (R1)            ;402=1
1978 006754  005002                    CLR     R2              ;R2=0
1979 006756  005012                    CLR     (R2)            ;0=0
1980 006760  005064  000376            CLR     376(R4)         ;400=0
1981 006764  001401                    BEQ     2$              ;BRANCH IF GOOD
1982 006766  104001            1$:     ERROR   +1              ;CPU ERROR
1983 006770  005364  000376      2$:     DEC     376(R4)         ;400=-1
1984 006774  005164  000400            COM     400(R4)         ;402=-1
1985 007000  001405                    BEQ     3$              ;BRANCH IF BAD
1986 007002  005264  000400            INC     400(R4)         ;402=-2
1987 007006  005264  000400            INC     400(R4)
1988 007012  001401                    BEQ     4$              ;BRANCH IF GOOD
1989                                                                   ;MODE 6 FAILED
1990 007014  104001            3$:     ERROR   +1              ;CPU ERROR
1991 007016  005261  177776      4$:     INC     -2(R1)          ;400=0
1992 007022  001401                    BEQ     6$              ;BRANCH IF GOOD
1993                                                                   ;INC MODE 6 FAILED
1994 007024  104001            5$:     ERROR   +1              ;CPU ERROR
1995 007026                    6$:
1996
1997                           ;
1999                           ;
2000                           ;
2002 007026                    MSPQ:
2003
2004                           ;       TEST NEG MODE 6
2005 007026  005001                    CLR     R1              ;R1=0
2006 007030  005004                    CLR     R4
2007 007032  105104                    COMB    R4
2008 007034  005204                    INC     R4              ;R4=400
2009 007036  005014                    CLR     (R4)            ;
2010 007040  005114                    COM     (R4)            ;400=-1
```

```
2011 007042  005044                    CLR     -(R4)           ;376=0
2012 007044  005044                    CLR     -(R4)           ;
2013 007046  005224                    INC     (R4)+           ;374=1 R4=376
2014 007050  005464  000002            NEG     2(R4)           ;400=1
2015 007054  001403                    BEQ     1$              ;NEGATE FAILED
2016 007056  102402                    BVS     1$
2017 007060  100401                    BMI     1$
2018 007062  103401                    BCS     2$              ;BRANCH IF GOOD
2019                                                           ;NEGATE FAILED
2020 007064  104001            1$:     ERROR   +1              ;CPU ERROR
2021 007066  005364  000002      2$:     DEC     2(R4)           ; TEST RESULT OF NEGATE
2022 007072  001401                    BEQ     4$              ;BRANCH IF GOOD
2023                                                           ;RESULT OF NEGATE FAILED
2024 007074  104001            3$:     ERROR   +1              ;CPU ERROR
2025 007076  005464  000000      4$:     NEG     0(R4)           ;*0=0
2026 007102  001401                    BEQ     5$              ; BRANCH IF GOOD
2027                                                           ;NEGATE FAILED
2028 007104  104001                    ERROR   +1              ;CPU ERROR
2029 007106  105461  000374      5$:     NEGB    374(R1)         ;374=0 377
2030 007112  102403                    BVS     6$              ;
2031 007114  001402                    BEQ     6$
2032 007116  100001                    BPL     6$
2033 007120  103401                    BCS     7$              ;BRANCH IF GOOD
2034                                                           ;NEGATE FAILED
2035 007122  104001            6$:     ERROR   +1              ;CPU ERROR
2036 007124  105261  000374      7$:     INCB    374(R1)         ;374=0
2037 007130  001401                    BEQ     9$              ;BRANCH IF GOOD
2038                                                           ;NEGATE FAILED
2039 007132  104001            8$:     ERROR   +1              ;CPU ERROR
2040 007134                    9$:
2041
2042                                   ;
2045 007134                    MSPR:
2046
2047                                   ;       TEST CLR, COM, INC - MODE 7
2048 007134  005001                            CLR     R1              ;R1=0
2049 007136  005004                            CLR     R4              ;
2050 007140  105104                            COMB    R4              ;
2051 007142  005204                            INC     R4              ;R4=400
2052 007144  005011                            CLR     (R1)
2053 007146  105111                            COMB    (R1)
2054 007150  005211                            INC     (R1)
2055 007152  005211                            INC     (R1)
2056 007154  005211                            INC     (R1)            ;;0=402
2057 007156  005014                            CLR     (R4)            ;400=0
2058 007160  005064  000002                    CLR     2(R4)           ;
2059 007164  005164  000002                    COM     2(R4)           ;402=-1
2060 007170  005074  177400                    CLR     @-400(R4)       ;402=0
2061 007174  001401                            BEQ     2$              ;BRANCH IF GOOD
2062 007176  104001            1$:             ERROR   +1              ;CPU ERROR
2063                                                                   ;INSTRUCTION FAILED
2064 007200  005171  000000      2$:             COM     @0(R1)          ;402=-1
2065 007204  100401                            BMI     4$              ;BRANCH IF GOOD
2066 007206  104001            3$:             ERROR   +1              ;CPU ERROR
2067
2068 007210  005104            4$:             COM     R4
2069 007212  005274  000401                    INC     @401(R4)                ;402=0
```

BASE INSTRUCTION SET TESTS

```
2070 007216  001401                        BEQ     6$              ;BRANCH IF GOOD
2071 007220  104001              5$:       ERROR   +1              ;CPU ERROR
2072                                                               ;MODE 7 FAILED
2073 007222                      6$:
2074
2075                             ;
2078 007222                      MSPS:
2079
2080                             ;       TEST NEG MODE 7
2081 007222  005004                      CLR     R4
2082 007224  005014                      CLR     (R4)            ;0=0
2083 007226  005002                      CLR     R2              ;
2084 007230  105102                      COMB    R2
2085 007232  005202                      INC     R2              ;R2=400
2086 007234  005012                      CLR     (R2)            ;400=0
2087 007236  005472  177400              NEG     @-400(R2)               ;NEG OF 0=0
2088 007242  103401                      BCS     1$              ;****
2089 007244  001401                      BEQ     2$              ;BRANCH IF GOOD
2090 007246  104001              1$:     ERROR   +1              ;CPU ERROR
2091                                                             ;
2092 007250  005314              2$:     DEC     (R4)            ;0=-1
2093 007252  005474  000400              NEG     @+400(R4)               ;0=1
2094 007256  001403                      BEQ     3$              ;BRANCH IF ERROR
2095 007260  102402                      BVS     3$              ;
2096 007262  100401                      BMI     3$              ;
2097 007264  103401                      BCS     4$              ;BRANCH IF GOOD
2098 007266  104001              3$:     ERROR   +1              ;CPU ERROR
2099                                                             ;NEGATE MODE 7 FAILED
2100 007270                      4$:
2101
2102                             ;
2105 007270                      MSPT:
2106
2107                             ;       TEST SINGLE OPERAND MODE 2 REG 7
2108 007270  005004                      CLR     R4
2109 007272  105104                      COMB    R4
2110 007274  005204                      INC     R4              ;R4=400
2111 007276  005027                      CLR     (R7)+           ;CLEAR NEXT LOCATION
2112 007300  177777              1$:     .WORD   -1              ;SETUP INITIAL DATA
2113 007302  001401                      BEQ     3$              ;BRANCH IF GOOD
2114 007304  104001              2$:     ERROR   +1              ;CPU ERROR
2115 007306                      3$:
2116
2117                             ;
2119
2121 007306                      MSPU:
2122
2123                             ;       TEST TST MODE 0
2124 007306  005004                      CLR     R4              ;R4=0
2125 007310  000277                      SCC                     ;CONDITION CODES =1111
2126 007312  000244                      CLZ                     ;CC=1011
2127 007314  005704                      TST     R4              ;*TEST INSTRUCTION
2128 007316  103403                      BCS     1$
2129 007320  102402                      BVS     1$
2130 007322  100401                      BMI     1$              ;BRANCH IF ERROR
2131 007324  001401                      BEQ     2$              ;BRANCH IF GOOD
2132 007326  104001              1$:     ERROR   +1              ;CPU ERROR
```

```
                                                        ;TST MODE 0 FAILED
007330  005304          2$:     DEC     R4              ;R4=-1
007332  000277                  SCC
007334  000250                  CLN                     ;CC=0111
007336  005704                  TST     R4              ;*TEST INSTRUCTION MODE 0
007340  103403                  BCS     3$              ;BRANCH IF ERROR
007342  102402                  BVS     3$
007344  001401                  BEQ     3$
007346  100401                  BMI     4$              ;BRANCH IF GOOD
007350  104001          3$:     ERROR   +1              ;CPU ERROR
                                                        ;TST FAILED
007352                  4$:
                        ;
                        MSPVO:
                        ;       TEST TST MODE 0 BYTE
007352  005004                  CLR     R4
007354  105104                  COMB    R4              ;0=000 377
007356  000277                  SCC
007360  000250                  CLN                     ;CC=0111
007362  105704                  TSTB    R4              ;*TEST INSTRUCTION ON EVEN BYTE
007364  102403                  BVS     1$              ;BRANCH IF ERROR
007366  103402                  BCS     1$              ;
007370  102401                  BVS     1$              ;
007372  100401                  BMI     2$              ;BRANCH IF GOOD
007374  104001          1$:     ERROR   +1              ;CPU ERROR
                                                        ;
007376  005204          2$:     INC     R4              ;POINT TO 1
007400  105704                  TSTB    R4              ; TEST INSTRUCTION
007402  001401                  BEQ     4$              ;BRANCH IF GOOD
007404  104001          3$:     ERROR   +1              ;CPU ERROR
                                                        ;TST FAILED ON BYTE
007406                  4$:
                        ;
                        MSPV:
                        ;       TEST TST MODE 1
007406  005004                  CLR     R4
007410  005014                  CLR     (R4)            ;0=0
007412  000277                  SCC
007414  000244                  CLZ                     ;CC=1011
007416  005714                  TST     (R4)            ;*TEST INSTRUCTION IN MODE 1
007420  103403                  BCS     1$              ;BRANCH IF ERROR
007422  102402                  BVS     1$
007424  100401                  BMI     1$              ;
007426  001401                  BEQ     2$              ;BRANCH IF GOOD
007430  104001          1$:     ERROR   +1              ;CPU ERROR

007432  005214          2$:     INC     (R4)            ;0=1
007434  000277                  SCC
007436  005714                  TST     (R4)            ; TEST INSTRUCTION
007440  001403                  BEQ     3$              ;BRANCH IF ERROR
007442  102402                  BVS     3$
007444  103401                  BCS     3$
007446  100001                  BPL     4$              ;BRANCH IF GOOD
```

```
2194 007450  104001                3$:     ERROR   +1              ;CPU ERROR
2195                                                                ;TST FAILED MODE 1
2196 007452                        4$:
2197
2198                               ;
2201 007452                        MSPX:
2202
2203                               ;       TEST TST MODE 1 BYTE
2204 007452  005004                        CLR     R4              ;R4=0
2205 007454  005014                        CLR     (R4)
2206 007456  105114                        COMB    (R4)
2207 007460  005214                        INC     (R4)            ;0=001 000
2208 007462  000277                        SCC
2209 007464  000244                        CLZ                     ;CC=1011
2210 007466  105714                        TSTB    (R4)            ;*TEST INTRUCTION
2211 007470  103403                        BCS     1$              ;BRANCH IF ERROR
2212 007472  102402                        BVS     1$              ;
2213 007474  100401                        BMI     1$
2214 007476  001401                        BEQ     2$              ;BRANCH IF GOOD
2215 007500  104001                1$:     ERROR   +1              ;CPU ERROR
2216
2217 007502  005204                2$:     INC     R4              ;R4=1
2218 007504  000277                        SCC
2219 007506  105714                        TSTB    (R4)            ; TEST INSTRUCTION
2220 007510  001403                        BEQ     3$              ;BRANCH IF ERROR
2221 007512  100402                        BMI     3$
2222 007514  102401                        BVS     3$
2223 007516  103001                        BCC     4$              ;BRANCH IF GOOD
2224 007520  104001                3$:     ERROR   +1              ;CPU ERROR
2225                                                               ;
2226 007522                        4$:
2227
2228                               ;
2231 007522                        MSPY:
2232
2233                               ;       TEST TST MODE 2
2234 007522  005004                        CLR     R4              ;
2235 007524  005024                        CLR     (R4)+           ;0=0
2236 007526  005014                        CLR     (R4)
2237 007530  005114                        COM     (R4)            ;2=-1
2238 007532  005004                        CLR     R4              ;R4=0
2239 007534  000277                        SCC             ;
2240 007536  000244                        CLZ             ;CC=1011
2241 007540  005724                        TST     (R4)+           ; TEST INSTRUCTION
2242 007542  103403                        BCS     1$              ;BRANCH IF ERROR
2243 007544  102402                        BVS     1$
2244 007546  100401                        BMI     1$
2245 007550  001401                        BEQ     2$              ;BRANCH IF GOOD
2246 007552  104001                1$:     ERROR   +1              ;CPU ERROR
2247                                                               ;MODE 2 TEST FAILED
2248 007554  005724                2$:     TST     (R4)+           ;TST LOC2
2249 007556  103403                        BCS     3$
2250 007560  102402                        BVS     3$
2251 007562  001401                        BEQ     3$
2252 007564  100401                        BMI     4$
2253 007566  104001                3$:     ERROR   +1              ;CPU ERROR
2254                                                               ;MODE 2 FAILED
```

```
2255 007570                        4$:
2256
2257                               ;
2260 007570                        MSPZ:
2261
2262                               ;       TEST TST MODE 2 BYTE
2263 007570  005004                        CLR     R4
2264 007572  005024                        CLR     (R4)+           ;
2265 007574  105144                        COMB    -(R4)           ;0=377 000
2266 007576  005304                        DEC     R4              ;R4=0
2267 007600  000277                        SCC
2268 007602  000244                        CLZ                     ;CC=1011
2269 007604  105724                        TSTB    (R4)+           ;
2270 007606  102403                        BVS     1$              ;BRANCH IF ERROR
2271 007610  103402                        BCS     1$
2272 007612  100401                        BMI     1$
2273 007614  001401                        BEQ     2$              ;BRANCH IF GOOD
2274 007616  104001                1$:     ERROR   +1              ;CPU ERROR
2275                                                               ;MODE 2 EVEN BYTE FAILED
2276 007620  000277                2$:     SCC
2277 007622  000250                        CLN                     ;CC=0111
2278 007624  105724                        TSTB    (R4)+           ;
2279 007626  001403                        BEQ     3$              ;
2280 007630  103402                        BCS     3$
2281 007632  102401                        BVS     3$
2282 007634  100401                        BMI     4$              ;BRANCH IF GOOD
2283 007636  104001                3$:     ERROR   +1              ;CPU ERROR
2284                                                               ;MODE 2 ODD BYTE FAILED
2285 007640                        4$:
2286
2287                               ;
2290 007640                        MSPAA:
2291
2292                               ;       TEST TST MODE 3
2293 007640  005004                        CLR     R4
2294 007642  005014                        CLR     (R4)
2295 007644  105114                        COMB    (R4)
2296 007646  005214                        INC     (R4)            ;0=400
2297 007650  005034                        CLR     @(R4)+          ;400=0
2298 007652  005004                        CLR     R4              ;R4=0
2299 007654  000277                        SCC
2300 007656  000244                        CLZ                     ;CC=1011
2301 007660  005734                        TST     @(R4)+          ; TEST MODE 3
2302 007662  103403                        BCS     1$              ;BRANCH IF ERROR
2303 007664  102402                        BVS     1$              ;
2304 007666  100401                        BMI     1$
2305 007670  001401                        BEQ     2$              ;BRANCH IF GOOD
2306 007672  104001                1$:     ERROR   +1              ;CPU ERROR
2307                                                               ;MODE 3 FAILED
2308 007674  005304                2$:     DEC     R4
2309 007676  005304                        DEC     R4              ;R4=0
2310 007700  005334                        DEC     @(R4)+          ;400=-1
2311 007702  005004                        CLR     R4
2312 007704  000277                        SCC
2313 007706  000250                        CLN                     ;CC=0111
2314 007710  005734                        TST     @(R4)+          ; TEST INSTRUCTION
2315 007712  103403                        BCS     3$              ;
```

```
 2316 007714  001402                    BEQ     3$              ;BRANCH IF ERROR
 2317 007716  102401                    BVS     3$              ;
 2318 007720  100401                    BMI     4$              ;BRANCH IF GOOD
 2319                                                           ;ERROR MODE 3
 2320 007722  104001            3$:     ERROR   +1              ;CPU ERROR
 2321 007724                    4$:
 2322
 2323                           ;
 2326 007724                    MSPBB:
 2327
 2328                           ;       TEST TST MODE 3 AUTO-INC
 2329 007724  005004                    CLR     R4
 2330 007726  005014                    CLR     (R4)            ;0=0
 2331 007730  105114                    COMB    (R4)            ;
 2332 007732  005214                    INC     (R4)            ;0=400
 2333 007734  005001                    CLR     R1
 2334 007736  105101                    COMB    R1
 2335 007740  005201                    INC     R1              ;R1=400
 2336 007742  005011                    CLR     (R1)            ;400=0
 2337 007744  000277                    SCC
 2338 007746  005734                    TST     @(R4)+          ;400=0
 2339 007750  103403                    BCS     1$              ;ERROR IF CARRY
 2340 007752  102402                    BVS     1$              ;ERROR IF OVERFLOW
 2341 007754  100401                    BMI     1$              ;ERROR IF MINUS
 2342 007756  001401                    BEQ     2$              ;ERROR IF NOT EQUAL
 2343 007760  104001            1$:     ERROR   +1              ;CPU ERROR
 2344                                                   ;CC SHOULD = 0100
 2345 007762  005304            2$:     DEC     R4
 2346 007764  005304                    DEC     R4
 2347 007766  005704                    TST     R4              ;SEE IF AUTO-INC WORKED
 2348 007770  001401                    BEQ     4$              ;ERROR IF R4 NE 0
 2349 007772  104001            3$:     ERROR   +1              ;CPU ERROR
 2350                                                   ;AUTO-INC FIALED
 2351 007774                    4$:
 2352
 2353                           ;
 2356 007774                    MSTB3:
 2357
 2358                           ;       TEST TST MODE 3 BYTE
 2359 007774  005004                    CLR     R4
 2360 007776  005014                    CLR     (R4)
 2361 010000  105114                    COMB    (R4)
 2362 010002  005214                    INC     (R4)
 2363 010004  005214                    INC     (R4)            ;0=401
 2364 010006  005001                    CLR     R1
 2365 010010  105101                    COMB    R1
 2366 010012  005201                    INC     R1              ;R1=400
 2367 010014  005011                    CLR     (R1)
 2368 010016  005111                    COM     (R1)
 2369 010020  105011                    CLRB    (R1)            ;400=377 000
 2370 010022  105734                    TSTB    @(R4)+          ;** TEST INSTRUCTION
 2371 010024  001403                    BEQ     1$              ;ERROR IF EQUAL
 2372 010026  103402                    BCS     1$              ;ERROR IF CARRY SET
 2373 010030  102401                    BVS     1$              ;ERROR IR OVERFLOW
 2374 010032  100401                    BMI     2$              ;BRANCH IF MINUS
 2375 010034  104001            1$:     ERROR   +1              ;CPU ERROR
 2376                                                   ;CC ERROR
```

BASE INSTRUCTION SET TESTS

```
2377 010036  005304                2$:     DEC     R4
2378 010040  005304                        DEC     R4
2379 010042  001401                        BEQ     4$                      ;BRANCH IF AUTO-INC WORKED
2380 010044  104001                3$:     ERROR   .+1                     ;CPU ERROR
2381                                                                ;AUTO-INC FAILED
2382 010046                        4$:
2383
2384                               ;
2387 010046                        MST4:
2388
2389                               ;       TEST TST MODE 4
2390 010046  005004                        CLR     R4
2391 010050  005014                        CLR     (R4)                    ;0=0
2392 010052  005204                        INC     R4
2393 010054  005204                        INC     R4                      ;R4=2
2394 010056  000277                        SCC
2395 010060  000244                        CLZ                             ;CC=1011
2396 010062  005744                        TST     -(R4)                   ;**TEST INTRUCTION
2397 010064  103403                        BCS     1$                      ;ERROR IF CARRY
2398 010066  102402                        BVS     1$                      ;ERROR IF OVERFLOW
2399 010070  100401                        BMI     1$                      ;ERROR IF MINUS
2400 010072  001401                        BEQ     2$                      ;BRANCH IF GOOD
2401 010074  104001                1$:     ERROR   .+1                     ;CPU ERROR
2402                                                                ;CC WRONG
2403 010076  005704                2$:     TST     R4                      ;INSURE CORRECT AUTO-DEC
2404 010100  001401                        BEQ     4$                      ;BRANCH IF GOOD AUTO-DEC
2405                                                                       ;BAD AUTO-DEC
2406 010102  104001                3$:     ERROR   .+1                     ;CPU ERROR
2407 010104                        4$:
2408
2409                               ;
2412 010104                        MST4B:
2413
2414                               ;       TEST TST MODE 4 BYTE
2415 010104  005004                        CLR     R4
2416 010106  005014                        CLR     (R4)
2417 010110  005114                        COM     (R4)
2418 010112  105114                        COMB    (R4)                    ;0=377 000
2419 010114  000277                        SCC
2420 010116  005204                        INC     R4
2421 010120  005204                        INC     R4                      ;R4=2
2422 010122  105744                        TSTB    -(R4)                   ;**TEST INSTRUCTION
2423 010124  001403                        BEQ     1$                      ;ERROR IF EQUAL TO 0
2424 010126  103402                        BCS     1$                      ;ERROR IF CARRY
2425 010130  102401                        BVS     1$                      ;ERROR IF OVERFLOW
2426 010132  100401                        BMI     2$                      ;BRANCH IF MINUS
2427 010134  104001                1$:     ERROR   .+1                     ;CPU ERROR
2428                                                                ;CC SHOULD EQUAL 0100
2429 010136  105744                2$:     TSTB    -(R4)                   ;**TEST EVEN BYTE
2430 010140  001401                        BEQ     4$                      ;BRANCH IF GOOD
2431 010142  104001                3$:     ERROR   .+1                     ;CPU ERROR
2432                                                               ;CC SHOULD EQUAL 0100 AND R4=-1
2433 010144                        4$:
2434
2437 010144                        MST5:
2438
2439                               ;       TEST TST MODE 5
```

BASE INSTRUCTION SET TESTS

```
2440 010144  005004                    CLR     R4              ;
2441 010146  005024                    CLR     (R4)+           ;0=0, R4=2
2442 010150  000277                    SCC
2443 010152  000244                    CLZ                     ;CC=1011
2444 010154  005754                    TST     @-(R4)          ; TEST INSTRUCTION
2445 010156  103403                    BCS     1$              ;ERROR IF CARRY
2446 010160  102402                    BVS     1$              ;ERROR IF OVERFLOW
2447 010162  100401                    BMI     1$              ;ERROR IF MINUS
2448 010164  001401                    BEQ     2$              ;BRANCH IF GOOD
2449 010166  104001             1$:    ERROR   .1              ;CPU ERROR
2450                                                   ;CC WRONG, SHOULD = 0100
2451 010170  005704             2$:    TST     R4
2452 010172  001401                    BEQ     4$              ;BRANCH IF AUTO-DEC WORKED
2453 010174  104001             3$:    ERROR   .1              ;CPU ERROR
2454                                                   ;AUTO-DEC FAILED
2455 010176                     4$:
2456
2457                            ;
2460 010176                     MST5B:
2461
2462                            ;       TEST TST MODE 5 BYTE
2463 010176  005004                    CLR     R4
2464 010200  005014                    CLR     (R4)
2465 010202  105114                    COMB    (R4)
2466 010204  005214                    INC     (R4)            ;0=400
2467 010206  005034                    CLR     @(R4)+          ;400=0, R4=2
2468 010210  005154                    COM     @-(R4)          ;
2469 010212  105134                    COMB    @(R4)+          ;400=377 000 R4=2
2470 010214  105754                    TSTB    @-(R4)          ;**TEST INSTRUCTION
2471 010216  103403                    BCS     1$              ;ERROR IF CARRY
2472 010220  100402                    BMI     1$              ;ERROR IF MINUS
2473 010222  102401                    BVS     1$              ;ERROR IF OVERFLOW
2474 010224  001401                    BEQ     2$              ;BRANCH IF GOOD
2475 010226  104001             1$:    ERROR   .1              ;CPU ERROR
2476                                                   ;CC SHOULD = 0100
2477 010230  005224             2$:    INC     (R4)+           ;0=401
2478 010232  105754                    TSTB    @-(R4)          ;**TEST INSTRUCTION
2479 010234  100401                    BMI     4$              ;BRANCH IF GOOD
2480                                                   ;EVEN BYTE FAILURE
2481 010236  104001             3$:    ERROR   .1              ;CPU ERROR
2482 010240                     4$:
2483
2484                            ;
2487 010240                     MST6:
2488
2489                            ;       TEST TST MODE 6
2490 010240  005004                    CLR     R4
2491 010242  005014                    CLR     (R4)            ;0=0
2492 010244  105104                    COMB    R4
2493 010246  005204                    INC     R4              ;R4=400
2494 010250  005014                    CLR     (R4)
2495 010252  005114                    COM     (R4)            ;400=-1
2496 010254  005764  177400            TST     -400(R4)        ;**TEST LOCATION 0
2497 010260  103403                    BCS     1$              ;ERROR IF CARRY
2498 010262  102402                    BVS     1$              ;ERROR IF OVERFLOW
2499 010264  100401                    BMI     1$              ;ERROR IF MINUS
2500 010266  001401                    BEQ     2$              ;BRANCH IF ZERO
```

BASE INSTRUCTION SET TESTS

```
2501 010270  104001                  1$:     ERROR   ;1                ;CPU ERROR
2502                                                        ;CC ARE WRONG
2503 010272  005004                  2$:     CLR     R4                 ;
2504 010274  005764  000400                  TST     400(R4)            ;TST LOCATION 400
2505 010300  001401                          BEQ     3$                 ;ERROR IF EQUAL
2506 010302  100401                          BMI     4$                 ;BRANCH IF MINUS
2507 010304  104001                  3$:     ERROR   ;1                 ;CPU ERROR
2508                                                        ;CC ERROR
2509 010306                          4$:
2510
2511                                 ;
2514 010306                          MST7:
2515
2516                                 ;       TEST TST MODE 7
2517 010306  005004                          CLR     R4
2518 010310  005014                          CLR     (R4)
2519 010312  005124                          COM     (R4)+              ;0=-1
2520 010314  005014                          CLR     (R4)               ;2=0
2521 010316  005002                          CLR     R2                 ;R2=0
2522 010320  005004                          CLR     R4
2523 010322  105104                          COMB    R4
2524 010324  005204                          INC     R4                 ;R4=400
2525 010326  005014                          CLR     (R4)               ;400=0
2526 010330  005774  177402                  TST     @-376(R4)          ;**TEST LOCATION 0
2527 010334  103403                          BCS     1$                 ;ERROR IF CARRY
2528 010336  102402                          BVS     1$                 ;ERROR IF OVERFLOW
2529 010340  001401                          BEQ     1$                 ;ERROR IF ZERO
2530 010342  100401                          BMI     2$                 ;BRANCH IF GOOD
2531 010344  104001                  1$:     ERROR   ;1                 ;CPU ERROR
2532                                                        ;ERROR IN CC
2533 010346  005222                  2$:     INC     (R2)+              ;0=-2 R2=2
2534 010350  005774  177402                  TST     @-376(R4)          ;** CHECK CONTENTS OF LOCATION 2
2535 010354  100401                          BMI     3$                 ;ERROR IF MINUS
2536 010356  001401                          BEQ     4$                 ;BRANCH IF GOOD
2537 010360  104001                  3$:     ERROR   ;1                 ;CPU ERROR
2538                                                        ;CC SHOULD = 0100
2539 010362                          4$:
2540
2541                                 ;
2543                                 ;
2544                                 ;
2545                                 ;
2546                                 ;
2547                                 .SBTTL ************ DOUBLE OPERAND TESTS *****************
2548                                 ;
2550 010362                          MDM0:
2551
2552                                 ;       TEST MOVE MODE 0
2553 010362  005004                          CLR     R4                 ;R4=0
2554 010364  005001                          CLR     R1
2555 010366  005101                          COM     R1                 ;R1=-1
2556 010370  010104                          MOV     R1,R4              ;**TEST MOVE INSTRUCTION
2557 010372  001402                          BEQ     1$                 ;ERROR IF R4=0
2558 010374  102401                          BVS     1$                 ;ERROR IF OBERFLOW
2559 010376  100401                          BMI     2$                 ;BRANCH IF MINUS
2560 010400  104001                  1$:     ERROR   ;1                 ;CPU ERROR
2561                                                        ;CC SHOULD = 100-, R4 SHOULD=-1
```

```
********** DOUBLE OPERAND TESTS *****************

 2562 010402  005204                    2$:     INC     R4              ;R4 SHOULD =0
 2563 010404  001375                            BNE     1$              ;ERROR IF R4 NE 0
 2564
 2565                                    ;
 2568 010406                             MDAO:
 2569
 2570                                    ;       TEST ADD MODE 0
 2571 010406  005004                            CLR     R4              ;R4=0
 2572 010410  005001                            CLR     R1
 2573 010412  005101                            COM     R1              ;R1=-1
 2574 010414  060104                            ADD     R1,R4           ;** TEST ADD OF R1 TO R4
 2575 010416  001403                            BEQ     1$              ;ERROR IF ZERO
 2576 010420  103402                            BCS     1$              ;ERROR IF CARRY
 2577 010422  102401                            BVS     1$              ;ERROR IF OVERFLOW
 2578 010424  100401                            BMI     2$              ;BRANCH IF MINUS
 2579 010426  104001                    1$:     ERROR   +1              ;CPU ERROR
 2580                                                            ;CC SHOULD = 1000 , R4=-1
 2581 010430  005204                    2$:     INC     R4              ;R4 SHOULD =0
 2582 010432  001401                            BEQ     4$              ;BRANCH IF R4=0
 2583 010434  104001                    3$:     ERROR   +1              ;CPU ERROR
 2584                                                            ;R4 SHOULD = 0
 2585 010436                            4$:
 2586
 2587                                    ;
 2590 010436                             MDSO:
 2591
 2592                                    ;       TEST SUB MODE 0
 2593 010436  005004                            CLR     R4
 2594 010440  005001                            CLR     R1
 2595 010442  005201                            INC     R1                      ;R1=1 R4=0
 2596 010444  160104                            SUB     R1,R4                   ;**TEST OF R4-R1, R4=-1
 2597 010446  102403                            BVS     1$                      ;ERROR IF V SET
 2598 010450  103002                            BCC     1$                      ;ERROR IF NO CARRY
 2599 010452  001401                            BEQ     1$                      ;ERROR IF =0
 2600 010454  100401                            BMI     2$                      ;BRANCH IF MINUS
 2601 010456  104001                    1$:     ERROR   +1              ;CPU ERROR
 2602                                                            ;CC SHOULD = 1001
 2603 010460  005101                    2$:     COM     R1                      ;R1=1
 2604 010462  005201                            INC     R1                      ;GET TWO'S COMPLIMENT, R1=-1
 2605 010464  160104                            SUB     R1,R4                   ;**TEST R4-R1 (1- 1= 0
 2606 010466  001401                            BEQ     4$                      ;BRANCH IF ZERO
 2607 010470  104001                    3$:     ERROR   +1              ;CPU ERROR
 2608                                                            ;CC SHOULD = 0100
 2609 010472                            4$:
 2610
 2611                                    ;
 2614 010472                             MDM27:
 2615
 2616                                    ;       TEST MOV MODE 27,00
 2617 010472  000257                            CCC                             ;CC=0000
 2618 010474  012704  125252                    MOV     #125252,R4              ;**TEST MOVE
 2619 010500  001401                            BEQ     1$                      ;ERROR IF = 0
 2620 010502  100401                            BMI     2$                      ;BRANCH IF MINUS
 2621 010504  104001                    1$:     ERROR   +1              ;CPU ERROR
 2622                                                            ;CC SHOULD = 1000
 2623 010506  012701  052525            2$:     MOV     #052525,R1              ;**TEST MOVE
 2624 010512  100401                            BMI     3$                      ;ERROR IF MINUS
```

```
********** DOUBLE OPERAND TESTS *****************

 2625 010514  001001                    BNE     4$              ;BRANCH IF NE 0
 2626 010516  104001            3$:     ERROR   +1      ;CPU ERROR
 2627                                                   ;CC SHOULD = 0000
 2628 010520  060104            4$:     ADD     R1,R4           ;R1+R4=-1
 2629 010522  100401                    BMI     6$              ;BRANCH IF MINUS
 2630 010524  104001            5$:     ERROR   +1      ;CPU ERROR
 2631                                                   ;MOV FAILED
 2632 010526  005204            6$:     INC     R4              ;R4+1=0
 2633 010530  001375                    BNE     5$              ;ERROR IF NOT ZERO
 2634
 2635                           ;
 2638 010532                    MBI00:
 2639
 2640                           ;       TEST BIC, BIS MODE 0,0
 2641 010532  005004                    CLR     R4
 2642 010534  005104                    COM     R4              ;R4=-1
 2643 010536  012701  125252            MOV     #125252,R1      ;SETUP R1 TEST DATA
 2644 010542  012702  052525            MOV     #052525,R2      ;R2=COMPLIMENT OF R1
 2645 010546  000261                    SEC
 2646 010550  040104                    BIC     R1,R4           ;**TEST BIC WITH CARRY SET
 2647 010552  103003                    BCC     1$              ;ERROR IF NO CARRY
 2648 010554  102402                    BVS     1$              ;ERROR IF OVERFLOW
 2649 010556  001401                    BEQ     1$              ;ERROR IF 0
 2650 010560  100001                    BPL     2$              ;BRANCH IF PLUS
 2651 010562  104001            1$:     ERROR   +1      ;CPU ERROR
 2652                                                   ;CC SHOULD = 0001
 2653 010564  020402            2$:     CMP     R4,R2           ;COMPARE CONTENTS OF R4 AND R2
 2654 010566  001401                    BEQ     4$              ;BRANCH IF EQUAL
 2655 010570  104001            3$:     ERROR   +1      ;CPU ERROR
 2656                                                   ;R4 AND R2 SHOULD BE EQUAL
 2657 010572  005301            4$:     DEC     R1              ;R1=125251
 2658 010574  050201                    BIS     R2,R1           ;BIS 052525 AND 125251=177775
 2659 010576  100401                    BMI     6$              ;BRANCH IF MIUS VALUE
 2660 010600  104001            5$:     ERROR   +1      ;CPU ERROR
 2661                                                   ;BAD BIS OPERATION
 2662 010602  005201            6$:     INC     R1
 2663 010604  005201                    INC     R1
 2664 010606  005201                    INC     R1              ;R1=0
 2665 010610  001373                    BNE     5$              ;ERROR IF NE 0
 2666
 2667                           ;
 2670 010612                    MBC00:
 2671
 2672                           ;       TEST BIT, CMP MODE 0,0
 2673 010612  012701  125252            MOV     #125252,R1      ;R1=125252
 2674 010616  012704  100000            MOV     #100000,R4      ;R4=100000
 2675 010622  012702  052525            MOV     #052525,R2      ;R2=052525
 2676 010626  030401                    BIT     R4,R1           ;**TEST OF BIT ,CC=1000
 2677 010630  001401                    BEQ     1$              ;ERROR IF EQ 0
 2678 010632  100401                    BMI     2$              ;BRANCH IF GOOD
 2679 010634  104001            1$:     ERROR   +1      ;CPU ERROR
 2680                                                   ;CC SHOULD = 1000
 2681 010636  020401            2$:     CMP     R4,R1           ;*TEST 100000-125252=25252
 2682 010640  001402                    BEQ     3$              ;ERROR IF EQUAL 0
 2683 010642  103001                    BCC     3$              ;ERROR IF CARRY CLEARED
 2684 010644  100401                    BMI     4$              ;BRANCH IF GOOD
 2685 010646  104001            3$:     ERROR   +1      ;CPU ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

 2686
 2687 010650  020104                4$:    CMP     R1,R4           ;CC SHOULD = 0010
 2688 010652  001403                       BEQ     5$                      ;125252-100000 = 25252
 2689 010654  103402                       BCS     5$              ;ERROR IF EQUAL
 2690 010656  102401                       BVS     5$              ;ERROR IF CARRY
 2691 010660  100001                       BPL     6$              ;ERROR IF OVERFLOW
 2692 010662  104001                5$:    ERROR   +1              ;BRANCH IF GOOD
 2693                                                               ;CPU ERROR
 2694 010664  005004                6$:    CLR     R4              ;CC SHOULD =0001
 2695 010666  005204                       INC     R4
 2696 010670  000277                       SCC                             ;R4=1
 2697 010672  030401                       BIT     R4,R1
 2698 010674  001401                       BEQ     8$                      ;R4 ↑ R1 = 2
 2699 010676  104001                7$:    ERROR   +1              ;BRANCH IF GOOD
 2700                                                               ;CPU ERROR
 2701 010700                        8$:                             ;CC SHOULD = 0101
 2702
 2703                               ;
 2706 010700                        MM11:
 2707
 2708                               ;      TEST MOV, MOVB MODE 1,1 AND SIGN EXT ON MOVB TO GPR
 2709 010700  012704  000400               MOV     #400,R4         ;R4=400
 2710 010704  012701  000402               MOV     #402,R1         ;R1=402
 2711 010710  005014                       CLR     (R4)            ;
 2712 010712  005114                       COM     (R4)            ;400=-1
 2713 010714  005011                       CLR     (R1)                    ;
 2714 010716  105111                       COMB    (R1)            ;402=000 377
 2715 010720  005002                       CLR     R2              ;R2=0
 2716 010722  012703  000405               MOV     #405,R3         ;R3=405
 2717 010726  000277                       SCC                     ;CC=1111
 2718 010730  011412                       MOV     (R4),(R2)       ;MOV    400 TO 0 ,0=-1
 2719 010732  001403                       BEQ     1$              ;ERROR IF 0
 2720 010734  102402                       BVS     1$              ;ERROR IF OVERFLOW
 2721 010736  103001                       BCC     1$              ;ERROR IF NO CARRY
 2722 010740  100401                       BMI     2$              ;BRANCH IF GOOD
 2723 010742  104001                1$:    ERROR   +1              ;CPU ERROR
 2724                                                       ;CC SHOULD =1001
 2725 010744  005212                2$:    INC     (R2)            ;0=0
 2726 010746  001004                       BNE     3$              ;ERROR IF NOT 0
 2727 010750  000257                       CCC                     ;CC=0000
 2728 010752  111113                       MOVB    (R1),(R3)               ;405=377
 2729 010754  001401                       BEQ     3$              ;ERROR IF EQUAL
 2730 010756  100401                       BMI     4$              ;BRANCH IF GOOD
 2731 010760  104001                3$:    ERROR   +1              ;CPU ERROR
 2732                                                       ;
 2733 010762  105213                4$:    INCB    (R3)            ;405=0
 2734 010764  001375                       BNE     3$              ;ERROR IF 405 NOT 0
 2735
 2736 010766  005002                ;CHECK THAT SIGN EXTENSION OCCURS ON A MOVB TO GENERAL REGISTER.
 2737 010770  111102                       CLR     R2              ;INIT R2 TO ZERO.
 2738 010772  100005                       MOVB    (R1),R2         ;MOVE 377 TO R2
 2739 010774  102404                       BPL     5$              ;ERROR! BIT 15 SHOULD BE SET.
 2740 010776  103403                       BVS     5$              ;V BIT SHOULD BE CLEARED
 2741 011000  022702  177777               BCS     5$              ;CARRY BIT SHOULD BE UNAFFECTED
 2742 011004  001401                       CMP     #177777,R2      ;TEST R2
 2743                                      BEQ     6$              ;SIGN EXTENDED THROUGH UPPER BYTE
 2744                                                                      ;ERROR! BYTE SHOULD HAVE
                                                                          ;SIGN EXTENDED THROUGH UPPER BYTE
```

```
********** DOUBLE OPERAND TESTS *****************

2745 011006  104001                  5$:     ERROR   +1              ;CPU ERROR
2746 011010                          6$:
2747
2748                                 ;
2751 011010                          MA11:
2752
2753                                 ;       TEST ADD MODE 1,1
2754 011010  012704  000400                  MOV     #400,R4         ;R4=400
2755 011014  012701  000402                  MOV     #402,R1         ;R1=402
2756 011020  012714  177753                  MOV     #-25,(R4)       ;400=-25
2757 011024  012711  000024                  MOV     #24,(R1)        ;402=24
2758 011030  061114                          ADD     (R1),(R4)       ;-25+24=-1
2759 011032  001404                          BEQ     1$              ;ERROR IF 0
2760 011034  103403                          BCS     1$              ;ERROR IF CARRY
2761 011036  100002                          BPL     1$              ;ERROR IF POSITIVE RESULT
2762 011040  005214                          INC     (R4)            ;-1+1=0
2763 011042  001401                          BEQ     2$              ;BRANCH IF GOOD
2764 011044  104001                  1$:     ERROR   +1              ;CPU ERROR
2765                                                        ;CC SHOULD = 1000
2766 011046                          2$:
2767
2768                                 ;
2771 011046                          MS11:
2772
2773                                 ;       TEST SUB MODE 1,1
2774 011046  012704  000400                  MOV     #400,R4         ;R4=400
2775 011052  012701  000404                  MOV     #404,R1         ;R1=404
2776 011056  012714  000003                  MOV     #3,(R4)         ;400=3
2777 011062  012711  000006                  MOV     #6,(R1)         ;406=6
2778 011066  000277                          SCC                     ;CC=1111
2779 011070  161411                          SUB     (R4),(R1)       ;6-3=3
2780 011072  001402                          BEQ     1$              ;ERROR IF 0
2781 011074  100401                          BMI     1$              ;ERROR IF MINUS
2782 011076  103001                          BCC     2$              ;BRANCH IF GOOD
2783 011100  104001                  1$:     ERROR   +1              ;CPU ERROR
2784                                                        ;CC SHOULD = 0000
2785 011102  161411                  2$:     SUB     (R4),(R1)       ;3-3=0
2786 011104  001375                          BNE     1$              ;ERROR IF NOT 0
2787
2788                                 ;
2791 011106                          MBB11:
2792
2793                                 ;       TEST BIC, BIS MODE 1,1
2794 011106  012704  000400                  MOV     #400,R4         ;R4=400
2795 011112  012701  000402                  MOV     #402,R1         ;R1=402
2796 011116  012714  052525                  MOV     #052525,(R4)    ;400=052525
2797 011122  012711  125252                  MOV     #125252,(R1)    ;402=125252
2798 011126  051411                          BIS     (R4),(R1)       ;R4 V R1 = -1
2799 011130  001401                          BEQ     1$              ;ERROR IF 0
2800 011132  100401                          BMI     2$              ;BRANCH IF GOOD
2801 011134  104001                  1$:     ERROR   +1              ;CPU ERROR
2802                                                        ;CC SHOULD = 1000
2803 011136  005211                  2$:     INC     (R1)            ;402=0
2804 011140  001401                          BEQ     4$              ;BRANCH IF GOOD
2805 011142  104001                  3$:     ERROR   +1              ;CPU ERROR
2806                                                        ;CC SHOULD = 0100
2807 011144  005311                  4$:     DEC     (R1)            ;402=-1
```

```
********** DOUBLE OPERAND TESTS *****************

 2808 011146  041411                    BIC     (R4),(R1)       ;R1=125252
 2809 011150  001401                    BEQ     5$              ;ERROR IF 0
 2810 011152  100401                    BMI     6$              ;BRANCH IF GOOD
 2811 011154  104001              5$:   ERROR   +1              ;CPU ERROR
 2812                                                 ;CC SHOULD = 1000
 2813 011156  005111              6$:   COM     (R1)            ;402=052525
 2814 011160  041114                    BIC     (R1),(R4)       ;400=0
 2815 011162  001401                    BEQ     8$              ;BRANCH IF GOOD
 2816 011164  104001              7$:   ERROR   +1              ;CPU ERROR
 2817                                                 ;CC SHOULD = 0100
 2818 011166                      8$:
 2819
 2820                             ;
 2823 011166                      MBC11:
 2824
 2825                             ;       TEST BIT, CMP MODE 1,1
 2826 011166  012704  000400               MOV     #400,R4                 ;R4=400
 2827 011172  012714  052525               MOV     #052525,(R4)            ;400=052525
 2828 011176  012701  000402               MOV     #402,R1                 ;R1-402
 2829 011202  012711  125252               MOV     #125252,(R1)            ;402=125252
 2830 011206  000241                       CLC                             ;CLEAR CARRY
 2831 011210  031411                       BIT     (R4),(R1)               ;**052525†125252=0
 2832 011212  103401                       BCS     1$                      ;ERROR IF CARRY
 2833 011214  001401                       BEQ     2$                      ;BRANCH IF GOOD
 2834 011216  104001              1$:      ERROR   +1                ;CPU ERROR
 2835                                                               ;CC SHOULD = 0100
 2836 011220  021411              2$:      CMP     (R4),(R1)               ;400-402=125253
 2837 011222  001403                       BEQ     3$                      ;ERROR IF ZERO
 2838 011224  103002                       BCC     3$                      ;ERROR IF NO CARRY
 2839 011226  102001                       BVC     3$                      ;ERROR IF NO OVERFLOW
 2840 011230  100401                       BMI     4$                      ;BRANCH IF GOOD
 2841 011232  104001              3$:      ERROR   +1                ;CPU ERROR
 2842                                                               ;CC SHOULD = 1000
 2843 011234  005014              4$:      CLR     (R4)
 2844 011236  005214                       INC     (R4)                    ;400=1
 2845 011240  031114                       BIT     (R1),(R4)               ;125252†1=0
 2846 011242  001401                       BEQ     6$                      ;BRANCH IF GOOD
 2847 011244  104001              5$:      ERROR   +1                ;CPU ERROR
 2848                                                               ;CC SHOULD= 0100
 2849 011246                      6$:
 2850
 2851                             ;
 2854 011246                      MM22:
 2855
 2856                             ;       TEST MOV MODE 2,2
 2857 011246  012704  000400               MOV     #400,R4         ;R4=400
 2858 011252  012701  000402               MOV     #402,R1         ;R1=402
 2859 011256  012714  000005               MOV     #5,(R4)         ;400=5
 2860 011262  005021                       CLR     (R1)+           ;402=0
 2861 011264  005011                       CLR     (R1)            ;
 2862 011266  005111                       COM     (R1)            ;404=-1
 2863 011270  005741                       TST     -(R1)           ;R1=402
 2864 011272  000277                       SCC                     ;CC=1111
 2865 011274  012124                       MOV     (R1)+,(R4)+     ;400=0 R4=402 R1=404
 2866 011276  100403                       BMI     1$              ;ERROR IF MINUS
 2867 011300  103002                       BCC     1$              ;ERROR IF NO CARRY
 2868 011302  102401                       BVS     1$              ;ERROR IF OVERFLOW
```

```
********** DOUBLE OPERAND TESTS *****************

 2869 011304  001401                    BEQ     2$              ;BRANCH IF GOOD
 2870 011306  104001            1$:     ERROR   +1              ;CPU ERROR
 2871                                                   ;CC SHOULD= 0101
 2872 011310  005244            2$:     INC     -(R4)           ;400=1 R4=400
 2873
 2874 011312  061411                    ADD     (R4),(R1)       ;1+ -1 =0
 2875 011314  001401                    BEQ     4$              ;BRANCH IF GOOD
 2876 011316  104001            3$:     ERROR   +1              ;CPU ERROR
 2877                                                   ;CC SHOULD = 0100
 2878 011320                    4$:
 2879
 2880                           ;
 2883 011320                    MS22:
 2884
 2885                           ;       TEST SUB MODE 2,2
 2886 011320  012704  000400            MOV     #400,R4                 ;R4=400
 2887 011324  012701  000402            MOV     #402,R1                 ;R1=402
 2888 011330  012714  177760            MOV     #177760,(R4)            ;400=177760
 2889 011334  012711  177750            MOV     #177750,(R1)            ;402=177750
 2890 011340  162421                    SUB     (R4)+,(R1)+             ;R1=177770
 2891 011342  001403                    BEQ     1$                      ;ERROR IF ZERO
 2892 011344  102402                    BVS     1$                      ;ERROR IF OVERFLOW
 2893 011346  103001                    BCC     1$                      ;ERROR IF NO CARRY
 2894 011350  100401                    BMI     2$                      ;BRANCH IF GOOD
 2895 011352  104001            1$:     ERROR   +1              ;CPU ERROR
 2896                                                   ;CC SHOULD=1000
 2897 011354  005241            2$:     INC     -(R1)                   ;R1=1777771
 2898 011356  162721  177771            SUB     #177771,(R1)+           ;R1=0
 2899 011362  100401                    BMI     3$                      ;ERROR IF MINUS
 2900 011364  001401                    BEQ     4$                      ;BRANCH IF GOOD
 2901 011366  104001            3$:     ERROR   +1              ;CPU ERROR
 2902                                                   ;CCSHOULD = 0100
 2903 011370                    4$:
 2904
 2905                           ;
 2921 011370                    MBB22:
 2922
 2923                           ;       TEST BIC, BICB, BIS, BISB MODE 2,2
 2924 011370  012704  000400            MOV     #400,R4         ;R4=400
 2925 011374  012701  000402            MOV     #402,R1         ;R1=402
 2926 011400  012702  000404            MOV     #404,R2         ;R2=404
 2927 011404  012714  141401            MOV     #141401,(R4)    ;400=303 001
 2928 011410  012711  177405            MOV     #177405,(R1)    ;402=377 005
 2929 011414  012722  000070            MOV     #70,(R2)+       ;404=2070
 2930 011420  012722  177777            MOV     #-1,(R2)+       ;406=-1
 2931 011424  042421                    BIC     (R4)+,(R1)+     ;402=074004
 2932 011426  001401                    BEQ     1$              ;ERROR IF ZERO
 2933 011430  100001                    BPL     2$              ;BRANCH IF GOOD
 2934                                                   ;CC SHOULD = 1000
 2935 011432  104001            1$:     ERROR   +1              ;CPU ERROR
 2936 011434  052421            2$:     BIS     (R4)+,(R1)+     ;404=074074
 2937 011436  142421                    BICB    (R4)+,(R1)+     ;406=074
 2938 011440  005301                    DEC     R1              ;R4=405 R1=406
 2939 011442  152421                    BISB    (R4)+,(R1)+     ;406=-1 R4=406 R1=407
 2940 011444  100401                    BMI     4$              ;BRANCH IF GOOD
 2941                                                   ;406 SHOULD=-1
 2942 011446  104001            3$:     ERROR   +1              ;CPU ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

 2943 011450  005214                  4$:     INC     (R4)            ;406 SHOULD=0
 2944 011452  001401                          BEQ     6$              ;BRANCH IF GOOD
 2945                                                                 ;ERROR! 406 NE 0
 2946 011454  104001                  5$:     ERROR   +1              ;CPU ERROR
 2947 011456                          6$:
 2948
 2949                                 ;
 2952 011456                          MBC22:
 2953
 2954                                 ;       TEST BIT, CMP MODE 2,2
 2955 011456  012704  000400                  MOV     #400,R4         ;R4=400
 2956 011462  012701  000402                  MOV     #402,R1         ;R1=402
 2957 011466  012714  125252                  MOV     #125252,(R4)    ;400=125252
 2958 011472  012721  100001                  MOV     #100001,(R1)+   ;402=100001
 2959 011476  012711  100002                  MOV     #100002,(R1)    ;404=100002
 2960 011502  005741                          TST     -(R1)           ;R1=402
 2961 011504  132421                          BITB    (R4)+,(R1)+     ;**ANDED RESULT= 000
 2962 011506  100401                          BMI     1$              ;ERROR IF MINUS
 2963 011510  001401                          BEQ     2$              ;BRANCH IF GOOD
 2964 011512  104001                  1$:     ERROR   +1              ;CPU ERROR
 2965                                                         ;CC SHOULD = 0100
 2966 011514  132124                  2$:     BITB    (R1)+,(R4)+     ;** ANDED RESULT = 200
 2967 011516  001401                          BEQ     3$              ;ERROR IF EQUAL
 2968 011520  100401                          BMI     4$              ;BRANCH IF GOOD
 2969 011522  104001                  3$:     ERROR   +1              ;CPU ERROR
 2970                                                 ;CC SHOULD= 1000 R4=402 R1=404
 2971 011524  022421                  4$:     CMP     (R4)+,(R1)+     ;RESULT =+1
 2972 011526  001402                          BEQ     5$              ;ERROR IF EQUAL
 2973 011530  103001                          BCC     5$              ;ERROR IF NO CARRY
 2974 011532  100401                          BMI     6$              ;BRANCH IF GOOD
 2975 011534  104001                  5$:     ERROR   +1              ;CPU ERROR
 2976                                                     ;CC SHOULD = 0000
 2977 011536  005341                  6$:     DEC     -(R1)           ;404=100001
 2978 011540  005741                          TST     -(R1)           ;R4=404 R1=402
 2979 011542  022124                          CMP     (R1)+,(R4)+     ;RESULT =0
 2980 011544  001401                          BEQ     8$              ;BRANCH IF GOOD
 2981                                                                 ;CC SHOULD = 0100 R1=404 R4=406
 2982 011546  104001                  7$:     ERROR   +1              ;CPU ERROR
 2983 011550                          8$:
 2984
 2985                                 ;
 2988 011550                          MS33:
 2989
 2990                                 ;       TEST SUB MODE 3,3
 2991 011550  005004                          CLR     R4              ;R4=0
 2992 011552  012701  000002                  MOV     #2,R1           ;R1=2
 2993 011556  012702  000400                  MOV     #400,R2         ;R2=400
 2994 011562  012714  000400                  MOV     #400,(R4)       ;0=400
 2995 011566  012711  000402                  MOV     #402,(R1)       ;2=402
 2996 011572  012722  000200                  MOV     #200,(R2)+      ;400=200
 2997 011576  012712  054320                  MOV     #54320,(R2)     ;402=54320
 2998 011602  163431                          SUB     @(R4)+,@(R1)+   ;54320 - 200=54120
 2999 011604  001402                          BEQ     1$              ;ERROR IF ZERO
 3000 011606  103401                          BCS     1$              ;ERROR IF CARRY
 3001 011610  100001                          BPL     2$              ;BRANCH IF GOOD
 3002 011612  104001                  1$:     ERROR   +1              ;CPU ERROR
 3003                                                     ;CC SHOULD =0001
```

```
********** DOUBLE OPERAND TESTS *****************

 3004 011614  022712  054120          2$:     CMP     #54120,(R2)     ; TEST R4 AUTO-INC AND RESULT
 3005 011620  001401                          BEQ     4$              ;BRANCH IF GOOD
 3006 011622  104001                  3$:     ERROR   +1              ;CPU ERROR
 3007                                                 ;CC SHOULD = 0100 R4=2 R1=4
 3008 011624  005067  166150          4$:     CLR     0               ;RESTORE VECTORS
 3009 011630  005067  166146                  CLR     2               ;       "
 3010
 3011
 3012                                 ;
 3015 011634                          MCB44:
 3016
 3017                                 ;       TEST CMP, BIT MODE 4,4
 3018 011634  012704  000400                  MOV     #400,R4                 ;R4=400
 3019 011640  012701  000402                  MOV     #402,R1                 ;R1=402
 3020 011644  012721  125366                  MOV     #125366,(R1)+           ;402=125366 R1=404
 3021 011650  012724  173001                  MOV     #173001,(R4)+           ;400=173001 R4=402
 3022 011654  024441                          CMP     -(R4),-(R1)             ;173001 - 125366=045603 CC=0000
 3023 011656  103401                          BCS     1$                      ;ERROR IF CARRY
 3024 011660  100001                          BPL     2$                      ;BRANCH IF GOOD
 3025                                                                         ;BAD COMPARE
 3026 011662  104001                  1$:     ERROR   +1              ;CPU ERROR
 3027 011664  005204                  2$:     INC     R4
 3028 011666  005201                          INC     R1                      ;R1=403
 3029 011670  000261                          SEC                             ;SET CARRY
 3030 011672  134144                          BITB    -(R1),-(R4)             ;173?1=0
 3031 011674  103001                          BCC     3$                      ;C SHOULD REMAIN SET
 3032 011676  001401                          BEQ     4$                      ;BRANCH IF GOOD
 3033                                                                         ;INCORRECT COMPARE R4=400 R1=402
 3034 011700  104001                  3$:     ERROR   +1              ;CPU ERROR
 3035 011702  005724                  4$:     TST     (R4)+                   ;R4=402
 3036 011704  005201                          INC     R1                      ;R1=403
 3037 011706  124441                          CMPB    -(R4),-(R1)             ;173 - 173=0
 3038 011710  001401                          BEQ     6$                      ;BRANCH IF GOOD
 3039                                                                         ;BAD COMPARE
 3040 011712  104001                  5$:     ERROR   +1              ;CPU ERROR
 3041 011714                          6$:
 3042                                 ;
 3045 011714                          MA55:
 3046
 3047                                 ;       TEST ADD MODE 5,5
 3048 011714  012704  000400                  MOV     #400,R4
 3049 011720  012724  000001                  MOV     #1,(R4)+                ;400=1
 3050 011724  012724  177776                  MOV     #-2,(R4)+               ;402=-2
 3051 011730  012724  000400                  MOV     #400,(R4)+              ;404=400
 3052 011734  012714  000402                  MOV     #402,(R4)               ;406=402 R4=406
 3053 011740  012701  000410                  MOV     #410,R1                 ;R1=410
 3054 011744  065451                          ADD     @-(R4),@-(R1)           ;1+ -2= -1
 3055 011746  001402                          BEQ     1$                      ;ERROR IF ZERO
 3056 011750  100001                          BPL     1$                      ;ERROR IF PLUS
 3057 011752  103001                          BCC     2$                      ;BRANCH IF GOOD
 3058 011754  104001                  1$:     ERROR   +1              ;CPU ERROR
 3059                                                                 ;BAD ADD
 3060 011756  062704  000004          2$:     ADD     #4,R4                   ;R4=410
 3061 011762  065154                          ADD     @-(R1),@-(R4)           ;-1 + 1 = 0
 3062 011764  001401                          BEQ     4$                      ;BRANCH IF GOOD
 3063 011766  104001                  3$:     ERROR   +1              ;CPU ERROR
 3064                                                                 ;CC SHOULD= 0100 R4=406 R1=402
```

```
********** DOUBLE OPERAND TESTS *****************

3065 011770                          4$:
3066
3067                                 ;
3070                                 ;
3071                                 ;
3072                                 ;---------------------------------
3073                                 ;TEST DOP BIT(B) MODE 6,6
3074                                 ;
3075                                       ;
3076                                       ;
3077                                       ;
3078 011770                          MB66:
3079
3080                                 ;     TEST BIT, BITB MODE 6,6
3081 011770  005004                        CLR     R4                      ;R4=0
3082 011772  012701  000400                MOV     #400,R1                 ;
3083 011776  012721  125252                MOV     #125252,(R1)+           ;400=125252
3084 012002  012721  000001                MOV     #1,(R1)+                ;402=1
3085 012006  012721  100000                MOV     #100000,(R1)+           ;404=1000000  R1=406
3086 012012  036461  000400  177774        BIT     400(R4),-4(R1)          ;(400)†(402)=0
3087 012020  001401                        BEQ     2$                      ;BRANCH IF GOOD
3088                                                                       ;CC SHOULD = 0100
3089 012022  104001                  1$:   ERROR   +1              ;CPU ERROR
3090 012024  136461  000405  177772  2$:   BITB    405(R4),-6(R1)          ;(405)†(400)=200
3091 012032  001401                        BEQ     3$                      ;ERROR IF ZERO
3092 012034  100401                        BMI     4$                      ;BRANCH IF GOOD
3093                                                                       ;CC SHOULD = 1000
3094 012036  104001                  3$:   ERROR   +1              ;CPU ERROR
3095 012040                          4$:
3096
3097                                 ;
3100 012040                          MS77:
3101
3102                                 ;     TEST SUB MODE 7,7
3103 012040  012704  000400                MOV     #400,R4         ;
3104 012044  005001                        CLR     R1
3105 012046  012724  177776                MOV     #-2,(R4)+               ;400=-2
3106 012052  012724  177777                MOV     #-1,(R4)+               ;402=-1
3107 012056  012724  000400                MOV     #400,(R4)+              ;404=400 R4=406
3108 012062  012711  000402                MOV     #402,(R1)               ;0=402
3109 012066  005201                        INC     R1                      ;R1=1
3110 012070  167471  177372  000403        SUB     @-406(R4),@403(R1)      ;-2 - -1 = -1
3111 012076  001401                        BEQ     1$                      ;ERROR  IF ZERO
3112 012100  100401                        BMI     2$                      ;BRANCH IF GOOD
3113                                                                       ;CC SHOULD=1000
3114 012102  104001                  1$:   ERROR   +1              ;CPU ERROR
3115 012104  167174  177777  177776  2$:   SUB     @-1(R1),@-2(R4)         ;-1 - -1 = 0
3116 012112  001401                        BEQ     4$                      ;BRANCH IF GOOD
3117 012114  104001                  3$:   ERROR   +1              ;CPU ERROR
3118                                                                       ;ERROR ON SUBTRACT 400=0
3119 012116  005067  165656          4$:   CLR     0                       ;RESTORE VECTORS
3120 012122  005067  165654                CLR     2                       ;       "
3121
3122
3123                                 ;
3126 012126                          MRL0:
3127
```

```
********** DOUBLE OPERAND TESTS *****************

3128
3129 012126  012704  125252          ;       TEST ROL, ROLB MODE 0
3130 012132  000277                          MOV     #125252,R4              ;R4=125252
3131 012134  006104                          SCC                             ;CC=1111
3132 012136  102004                          ROL     R4                      ;R4=052525 WITH CARRY SET
3133 012140  103003                          BVC     1$                      ;ERROR IF V CLEAR
3134 012142  022704  052525                  BCC     1$                      ;ERROR IF CARRY CLEAR
3135 012146  001401                          CMP     #052525,R4              ;SEE IF RO = EXPECTED
3136 012150  104001                          BEQ     2$                      ;ERROR IF R4 NE EXPECTD
3137                                 1$:     ERROR   +1              ;CPU ERROR
3138 012152  012704  125252                                          ;ROL FAILED, CC SHOULD=0011
3139 012156  000257                  2$:     MOV     #125252,R4              ;R4=125252
3140 012160  106104                          CCC                             ;CC=0000
3141 012162  103005                          ROLB    R4                      ;ROTATE EVEN BYTE
3142 012164  102004                          BCC     3$                      ;ERROR IF NO CARRY
3143 012166  100403                          BVC     3$                      ;ERROR IF NO OVERFLOW
3144 012170  022704  125124                  BMI     3$                      ;ERROR IF MINUS
3145 012174  001401                          CMP     #125124,R4              ;SEE IF R4 = EXPECTED
3146 012176  104001                          BEQ     4$                      ;BRANCH IF GOOD
3147                                 3$:     ERROR   +1              ;CPU ERROR
3148 012200                          4$:                             ;ROLB FAILED, CC SHOULD=1011, R4=125125
3149
3150                                 ;
3153 012200                          MRLB1:
3154
3155                                 ;       TEST ROL, ROLB MODE 1
3156 012200  005004                          CLR     R4                      ;R4=0
3157 012202  012714  052525                  MOV     #52525,(R4)             ;0=52525
3158 012206  006114                          ROL     (R4)                    ;**TEST INSTRUCTION, 0=125252
3159 012210  100005                          BPL     1$                      ;ERROR IF PLUS
3160 012212  102004                          BVC     1$                      ;ERROR IF NO OVERFLOW
3161 012214  103403                          BCS     1$                      ;ERROR IF CARRY
3162 012216  021427  125252                  CMP     (R4),#125252            ;SEE IF R4=EXPECTED
3163 012222  001401                          BEQ     2$                      ;BRANCH IF GOOD
3164                                                                         ;BAD ROL ,CC SHOULD=1010
3165 012224  104001                  1$:     ERROR   +1              ;CPU ERROR
3166 012226  012714  125252          2$:     MOV     #125252,(R4)            ;0=125252
3167 012232  005204                          INC     R4                      ;R4=1
3168 012234  000277                          SCC                             ;CC=1111
3169 012236  106114                          ROLB    (R4)                    ;**TEST INSTRUCTION
3170 012240  100406                          BMI     3$                      ;ERROR IF RESULT IS POSITIVE
3171 012242  103005                          BCC     3$                      ;ERROR IF NO CARRY
3172 012244  102004                          BVC     3$                      ;ERROR IF V CLEAR
3173 012246  005304                          DEC     R4                      ;R4=0
3174 012250  022714  052652                  CMP     #52652,(R4)             ;ERROR IF 0 NE EXPECTED
3175 012254  001401                          BEQ     4$                      ;BRANCH IF GOOD
3176 012256  104001                  3$:     ERROR   +1              ;CPU ERROR
3177                                                                 ;BAD ROLB ODD BYTE,CC SHOULD=1011
3178 012260                          4$:
3179
3180                                 ;
3183 012260                          MRL2:
3184
3185                                 ;       TEST ROL, ROLB MODE 2
3186 012260  005004                          CLR     R4                      ;R4=0
3187 012262  012714  100000                  MOV     #100000,(R4)            ;0=100000
3188 012266  000257                          CCC                             ;CC=0000
```

```
********** DOUBLE OPERAND TESTS ******************

 3189 012270  006124                        ROL     (R4)+                   ;*TEST INSTRUCTION
 3190 012272  103002                        BCC     1$                      ;ERROR IF NO CARRY
 3191 012274  102001                        BVC     1$                      ;ERROR IF NO OVERFLOW
 3192 012276  001401                        BEQ     2$                      ;BRANCH IF GOOD
 3193                                                                       ;ROL FAILED ,CCSHOULD= 0100
 3194 012300  104001                  1$:   ERROR   +1                ;CPU ERROR
 3195 012302  005304                  2$:   DEC     R4
 3196 012304  005304                        DEC     R4
 3197 012306  001012                        BNE     3$                      ;ERROR IN AUTO-DEC
 3198 012310  012714  004040                MOV     #4040,(R4)              ;0=4040
 3199 012314  000241                        CLC
 3200 012316  106124                        ROLB    (R4)+                   ;**TEST INSTURCTION
 3201 012320  103405                        BCS     3$                      ;ERROR IF CARRY SET
 3202 012322  102404                        BVS     3$                      ;ERROR IF V
 3203 012324  005304                        DEC     R4
 3204 012326  022714  004100                CMP     #04100,(R4)             ;SEE IF 0= EXPECTED RESULT
 3205 012332  001401                        BEQ     4$                      ;BRANCH IF GOOD
 3206 012334  104001                  3$:   ERROR   +1                ;CPU ERROR
 3207                                                                   ;BAD ROL
 3208 012336                          4$:
 3209
 3210                                 ;
 3213 012336                          MRL3:
 3214
 3215                                 ;       TEST ROL, ROLB MODE 3
 3216 012336  005004                        CLR     R4                      ;R4=0
 3217 012340  012714  052525                MOV     #052525,(R4)                    ;0=52525
 3218 012344  000277                        SCC                             ;CC=1111
 3219 012346  006137  000000                ROL     @#0                     ;**TEST INSRUCTION MODE 3 WITH PC
 3220 012352  100005                        BPL     1$                      ;ERROR IF PLUS
 3221 012354  102004                        BVC     1$                      ;ERROR IF NO OVERFLOW
 3222 012356  103403                        BCS     1$                      ;ERROR IF CARRY
 3223 012360  022714  125253                CMP     #125253,(R4)            ;COMPARE RESULT WITH EXPECTED
 3224 012364  001401                        BEQ     2$                      ;BRANCH IF GOOD
 3225                                                                       ;BAD ROL CC SHOULD=1010
 3226 012366  104001                  1$:   ERROR   +1                ;CPU ERROR
 3227 012370  012714  125252          2$:   MOV     #125252,(R4)            ;0=125252
 3228 012374  000261                        SEC                             ;CC=---1
 3229 012376  106137  000000                ROLB    @#0                     ;**TEST INSTRUCTION
 3230 012402  100402                        BMI     3$                      ;ERROR IF MINUS
 3231 012404  103001                        BCC     3$                      ;ERROR IF NO CARRY
 3232 012406  102401                        BVS     4$                      ;BRANCH IF OVERFLOW
 3233 012410  104001                  3$:   ERROR   +1                ;CPU ERROR
 3234                                                                   ;BAD ROL, CC SHOULD=1011
 3235 012412                          4$:
 3236
 3237                                 ;
 3240 012412                          MRL4:
 3241
 3242                                 ;       TEST ROL MODE 4
 3243 012412  005001                        CLR     R1                      ;R1=0
 3244 012414  012704  000002                MOV     #2,R4                   ;R4=2
 3245 012420  012711  054321                MOV     #54321,(R1)             ;0=54321
 3246 012424  000277                        SCC                             ;CC=1111
 3247 012426  006144                        ROL     -(R4)                   ;**TEST INSTRUCTION
 3248 012430  100007                        BPL     1$                      ;ERROR IF PLUS
 3249 012432  102006                        BVC     1$                      ;ERROR IF NO OVERFLOW
```

```
********** DOUBLE OPERAND TESTS *****************

3250 012434  103405                    BCS     1$              ;ERROR IF CARRY
3251 012436  022711  130643             CMP     #130643,(R1)    ;SEE IF EXPECTED RESULT
3252 012442  001002                    BNE     1$              ;BRANCH IF ROL FAILED
3253 012444  005704                    TST     R4              ;SEE IF AUTO-DEC WORKED
3254 012446  001401                    BEQ     2$              ;BRANCH IF GOOD
3255                                                           ;ERROR! BAD ROL INST
3256 012450  104001            1$:     ERROR   +1              ;CPU ERROR
3257 012452                    2$:
3258
3259                           ;
3262 012452                    MMRL5:
3263
3264                           ;       TEST ROL MODE 5
3265 012452  005004                    CLR     R4                      ;R4=0
3266 012454  012714  000400             MOV     #400,(R4)               ;0=400
3267 012460  012734  123456             MOV     #123456,@(R4)+          ;400=123465, R4=2
3268 012464  000277                     SCC                             ;CC=1111
3269 012466  006154                     ROL     @-(R4)                  ;**TEST INSTRUCTION
3270 012470  100410                     BMI     1$                      ;ERROR IF RESULT IS MINUS
3271 012472  103007                     BCC     1$                      ;ERROR IF NO CARRY
3272 012474  102006                     BVC     1$                      ;ERROR IF NO OVERFLOW
3273 012476  005704                     TST     R4                      ;SEE IF AUTO-DEC WORKED
3274 012500  001004                     BNE     1$                      ;ERROR OF AUTO-DEC
3275 012502  022737  047135  000400     CMP     #47135,@#400            ;SEE  IF CORRECT RESULT
3276 012510  001401                     BEQ     2$                      ;BRANCH IF GOOD
3277                                                                    ;BAD ROL MODE 5
3278 012512  104001            1$:     ERROR   +1              ;CPU ERROR
3279 012514                    2$:
3280
3281                           ;
3284 012514                    MRL6:
3285
3286                           ;       TEST ROL MODE 6
3287 012514  012704  000400             MOV     #400,R4                 ;R4=400
3288 012520  005001                     CLR     R1                      ;R1=0
3289 012522  012711  032525             MOV     #32525,(R1)             ;0=32525
3290 012526  000277                     SCC
3291 012530  006164  177400             ROL     -400(R4)                ;**TEST INSTRUCTION
3292 012534  100405                     BMI     1$                      ;ERROR IF MINUS
3293 012536  103404                     BCS     1$                      ;ERROR IF CARRY
3294 012540  102403                     BVS     1$                      ;ERROR IF OVERFLOW
3295 012542  022711  065253             CMP     #65253,(R1)             ;SEE IF CORRECT RESULT
3296 012546  001401                     BEQ     2$                      ;BRANCH IF GOOD
3297                                                                    ;BAD ROL MODE 6
3298 012550  104001            1$:     ERROR   +1              ;CPU ERROR
3299 012552                    2$:
3300
3301                           ;
3304 012552                    MRL7:
3305
3306                           ;       TEST ROL MODE 7
3307 012552  012704  000400             MOV     #400,R4                 ;R4=400
3308 012556  005037  000402             CLR     @#402                   ;402=0
3309 012562  012737  100000  000000     MOV     #100000,@#0             ;0=100000
3310 012570  006174  000002             ROL     @2(R4)                  ;**TEST INSTRUCTION
3311 012574  100406                     BMI     1$                      ;ERROR IF MINUS
3312 012576  001005                     BNE     1$                      ;ERROR IF NOT ZERO
```

```
********** DOUBLE OPERAND TESTS *****************

 3313 012600  103004                        BCC     1$                      ;ERROR IF NO CARRY
 3314 012602  102003                        BVC     1$                      ;ERROR IF NO OVERFLOW
 3315 012604  005737  000000                TST     @#0                     ;CHECK RESULT
 3316 012610  001401                        BEQ     2$                      ;BRANCH IF GOOD
 3317                                                                       ;BAD ROL MODE 7
 3318 012612  104001                1$:     ERROR   +1              ;CPU ERROR
 3319 012614                        2$:
 3320
 3321                                ;
 3382 012614                        MSW37:
 3383
 3384                                ;       TEST SWAB MODE 37
 3385 012614  012704  000400                MOV     #400,R4                 ;R4=400
 3386 012620  012714  040700                MOV     #40700,(R4)             ;400= 101 300
 3387 012624  000337  000400                SWAB    @#400                   ;400 SHOULD = 300 101
 3388 012630  100406                        BMI     1$                      ;ERROR IF MINUS
 3389 012632  022714  140101                CMP     #140101,(R4)            ;SEE IF EXPECTED RESULT
 3390 012636  001003                        BNE     1$                      ;BRANCH IF BAD
 3391 012640  000337  000400                SWAB    @#400                   ;400=101 300
 3392 012644  100401                        BMI     2$                      ;BRANCH IF GOOD
 3393                                                                       ;ERROR! BAD SWAB MODE 37
 3394 012646  104001                1$:     ERROR   +1              ;CPU ERROR
 3395 012650                        2$:
 3396
 3397                                ;
 3400 012650                        MRR0:
 3401
 3402                                ;       TEST ROR MODE 0
 3403 012650  012704  052525                MOV     #52525,R4               ;R4=52525
 3404 012654  000257                        CCC                             ;CC=0000
 3405 012656  006004                        ROR     R4                      ;R4 SHOULD = 25252 WITH CARRY
 3406 012660  103003                        BCC     1$                      ;ERROR IF NO CARRY
 3407 012662  022704  025252                CMP     #25252,R4               ;SEE IF R4= EXPECTED
 3408 012666  001401                        BEQ     2$                      ;BRANCH IF GOOD
 3409                                                                       ;ROR MODE 0 FAILED
 3410 012670  104001                1$:     ERROR   +1              ;CPU ERROR
 3411 012672                        2$:
 3412
 3413                                ;
 3416 012672                        MRRB1:
 3417
 3418                                ;       TEST RORB MODE 1
 3419 012672  005004                        CLR     R4                      ;R4=0
 3420 012674  012714  000001                MOV     #1,(R4)                 ;0=1
 3421 012700  000277                        SCC                             ;CC=1111
 3422 012702  106014                        RORB    (R4)                    ;0=000200, NO C
 3423 012704  103004                        BCC     1$                      ;ERROR IF NO CARRY
 3424 012706  100003                        BPL     1$                      ;ERROR IF PLUS
 3425 012710  022714  000200                CMP     #200,(R4)               ;CHECK RESULT
 3426 012714  001401                        BEQ     2$                      ;BRANCH IF GOOD
 3427 012716  104001                1$:     ERROR   +1              ;CPU ERROR
 3428                                                                       ;BAD RESULT
 3429 012720                        2$:
 3430
 3431                                ;
 3433
 3435 012720                        MJ:
```

```
********** DOUBLE OPERAND TESTS *****************

 3436
 3437                                    ;       TEST JMP - ALL MODES
 3438 012720  012737  000001  003072              MOV     #1,@#SEQ        ;SETUP TEST SEQUENCER
 3439 012726  012701  012772                      MOV     #MJU1,R1        ;SET MODE 1 JUMP ADDRESS
 3440 012732  000111                              JMP     (R1)            ;*JMP MODE 1
 3441 012734  023727  003072  000002  MJU2:      CMP     @#SEQ,#2        ;CHECK FOR CORRECT SEQUENCE
 3442 012742  001401                              BEQ     MJU2A           ;BRANCH IF GOOD
 3443                                                                      ;MODE 2 JUMP FAILED
 3444 012744  104001                      1$:     ERROR   +1      ;CPU ERROR
 3445 012746  020127  012736              MJU2A:  CMP     R1,#MJU2+2      ;CHECK FOR AUTO-INCREMENT
 3446 012752  001401                              BEQ     MJU2B           ;BRANCH IF GOOD
 3447 012754  104001                      2$:     ERROR   +1      ;CPU ERROR
 3448                                             ;AUTO-INCR FAILED
 3449 012756  005237  003072              MJU2B:  INC     @#SEQ           ;UPDATE TEST SEQUENCER
 3450 012762  012701  012770                      MOV     #MJ2,R1         ;SETUP MODE 3 JUMP
 3451 012766  000131                              JMP     @(R1)+          ;JUMP MODE 3
 3452 012770  013016                      MJ2:    .WORD   MJU3            ;MODED 3 DETINATION
 3453 012772  023727  003072  000001  MJU1:      CMP     @#SEQ,#1        ; TEST FOR CORRECT SEQUENCE
 3454 013000  001401                              BEQ     MJU1A           ;BRANCH IF GOOD
 3455 013002  104001                      3$:     ERROR   +1      ;CPU ERROR
 3456                                             ;JMP OUT OF SEQUENCE
 3457 013004  005237  003072              MJU1A:  INC     @#SEQ           ;UPDATE SEQUENCE
 3458 013010  012701  012734                      MOV     #MJU2,R1        ;SETUP MODE 2 DESTINATION
Z3459 013014  000121                              JMP     (R1)+           ;JUMP MODE 2
 3460 013016  023727  003072  000003  MJU3:      CMP     @#SEQ,#3        ; TEST FOR CORRECT SEQUENCE
 3461 013024  001401                              BEQ     MJU3A           ;BRANCH IF GOOD
 3462 013026  104001                      4$:     ERROR   +1      ;CPU ERROR
 3463                                             ;JMP OUT OF SEQUENCE
 3464 013030  022701  012772              MJU3A:  CMP     #MJ2+2,R1       ; TEST AUTO-INCREMENT
 3465 013034  001401                              BEQ     MJU3B           ;BRANCH IF GOOD
 3466 013036  104001                      5$:     ERROR   +1      ;CPU ERROR
 3467                                             ;AUTO-INCREMENT FAILED MODE 3
 3468 013040  005237  003072              MJU3B:  INC     @#SEQ           ;UPDATE SEQUENCER
 3469 013044  012701  013114                      MOV     #MJU4+2,R1      ;SETUP DESTINATION MODE 4
 3470 013050  000141                              JMP     -(R1)           ;EXECUTE JUMP MODE 4
 3471 013052  000000                      MJU5:   HALT
 3472 013054  022701  013150                      CMP     #MJ5,R1         ;CHECK AUTO-DECREMENT
 3473 013060  001401                              BEQ     MJU5A           ;BRANCH IF GOOD AUTO-DEC
 3474 013062  104001                      6$:     ERROR   +1      ;CPU ERROR
 3475                                             ;AUTO-DEC FAIELD MODE 5
 3476 013064  023727  003072  000005  MJU5A:  CMP     @#SEQ,#5        ; TEST CORRECT SEQUENCE
 3477 013072  001401                              BEQ     MJU5B           ;BRANCH IF GOOD SEQUENCE
 3478 013074  104001                      7$:     ERROR   +1      ;CPU ERROR
 3479                                             ;JMP OUT OF SEQUENCE
 3480 013076  005237  003072              MJU5B:  INC     @#SEQ           ;UPDATE SEQUENCE COUNT
 3481 013102  012701  013145                      MOV     #MJU6-5,R1      ;SETUP DESTINATION MODE6
 3482 013106  000161  000005                      JMP     +5(R1)          ;JUMP MODE 6
 3483                                     ;;
 3484 013112  000240                      MJU4:   NOP
 3485 013114  022701  013112                      CMP     #MJU4,R1        ; TEST AUTO-DECR
 3486 013120  001401                              BEQ     MJU4A           ;BRANCH IF GOOD
 3487 013122  104001                      8$:     ERROR   +1      ;CPU ERROR
 3488                                             ;MODE 4 AUTO-DEC FAILED
 3489 013124  023727  003072  000004  MJU4A:  CMP     @#SEQ,#4        ; TEST FOR CORRECT SEQUENCE
 3490 013132  001401                              BEQ     MJU4B           ;BRANCH IF CORRECT SEQUENCE
 3491 013134  104001                      9$:     ERROR   +1      ;CPU ERROR
 3492                                             ;INCORRECT JMP SEQUENCE
```

```
********** DOUBLE OPERAND TESTS *****************

 3493 013136  005237  003072          MJU4B:  INC     @#SEQ                   ;UPDATE SEQUENCE
 3494 013142  012701  013152                  MOV     #MJ5+2,R1               ;SETUP MODE 5 POINTER
 3495 013146  000151                          JMP     @-(R1)                  ;EXECUTE MODE 5 JMP
 3496                                 ;
 3497 013150  013054                  MJ5:    .WORD   MJU5+2                  ;POINTER MODE 5
 3498                                 ;
 3499 013152  022737  000006  003072  MJU6:    CMP     #6,@#SEQ                ;CHECK FOR CORRECT SEQUENCE
 3500 013160  001401                          BEQ     MJU6A                   ;BRANCH IF GOOD
 3501 013162  104001                  10$:    ERROR   +1              ;CPU ERROR
 3502                                         ;INCORRECT SEQUENCE
 3503 013164  005237  003072          MJU6A:  INC     @#SEQ                   ;UPDATE SEQUENCER
 3504 013170  012701  013210                  MOV     #MJ7+10,R1              ;SETUP INDEX
 3505 013174  000171  177770                  JMP     @-10(R1)                ;EXECUTE MODE 7 JUMP
 3506 013200  013204                  MJ7:    .WORD   MJU7                    ;POINTER FOR MODE 7
 3507 013202  000000                          HALT
 3508 013204  022737  000007  003072  MJU7:    CMP     #7,@#SEQ                ; TEST FOR CORRECT SEQUENCE
 3509 013212  001401                          BEQ     MJU7E                   ;BRANCH IF GOOD SEQUENCE
 3510 013214  104001                  11$:    ERROR   +1              ;CPU ERROR
 3511                                         ; TESTING OUT OF SEQUENCE
 3512 013216                          MJU7E:
 3513                                 ;
 3524
 3525                                 ;       TEST THAT PRE-FETCH BUFFER CAN BE OVER WRITTEN
 3526 013216  012701  013526                  MOV     #128$,R1                ;SET UP R1 WITH ADDRESS OF ERROR
 3527                                                                         ;ROUTINE
 3528         000040                  .REPT   32.
 3530 013222  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
      013224  000240                          .WORD   NOP                     ;NOP INSTRUCTION
      013226  000111                  30008$: .WORD   111                     ;JMP    (R1)
      013230  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
      013232  000240                          .WORD   NOP                     ;NOP INSTRUCTION
      013234  000111                  30009$: .WORD   111                     ;JMP    (R1)
      013236  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
      013240  000240                          .WORD   NOP                     ;NOP INSTRUCTION
      013242  000111                  30010$: .WORD   111                     ;JMP    (R1)
      013244  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
      013246  000240                          .WORD   NOP                     ;NOP INSTRUCTION
      013250  000111                  30011$: .WORD   111                     ;JMP    (R1)
      013252  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
      013254  000240                          .WORD   NOP                     ;NOP INSTRUCTION
      013256  000111                  30012$: .WORD   111                     ;JMP    (R1)
      013260  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
      013262  000240                          .WORD   NOP                     ;NOP INSTRUCTION
      013264  000111                  30013$: .WORD   111                     ;JMP    (R1)
      013266  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
      013270  000240                          .WORD   NOP                     ;NOP INSTRUCTION
      013272  000111                  30014$: .WORD   111                     ;JMP    (R1)
      013274  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
      013276  000240                          .WORD   NOP                     ;NOP INSTRUCTION
      013300  000111                  30015$: .WORD   111                     ;JMP    (R1)
      013302  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
      013304  000240                          .WORD   NOP                     ;NOP INSTRUCTION
      013306  000111                  30016$: .WORD   111                     ;JMP    (R1)
      013310  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
      013312  000240                          .WORD   NOP                     ;NOP INSTRUCTION
      013314  000111                  30017$: .WORD   111                     ;JMP    (R1)
      013316  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
```

********** DOUBLE OPERAND TESTS **********

```
013320  000240                  .WORD   NOP             ;NOP INSTRUCTION
013322  000111          30018$: .WORD   111             ;JMP    (R1)
013324  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013326  000240                  .WORD   NOP             ;NOP INSTRUCTION
013330  000111          30019$: .WORD   111             ;JMP    (R1)
013332  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013334  000240                  .WORD   NOP             ;NOP INSTRUCTION
013336  000111          30020$: .WORD   111             ;JMP    (R1)
013340  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013342  000240                  .WORD   NOP             ;NOP INSTRUCTION
013344  000111          30021$: .WORD   111             ;JMP    (R1)
013346  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013350  000240                  .WORD   NOP             ;NOP INSTRUCTION
013352  000111          30022$: .WORD   111             ;JMP    (R1)
013354  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013356  000240                  .WORD   NOP             ;NOP INSTRUCTION
013360  000111          30023$: .WORD   111             ;JMP    (R1)
013362  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013364  000240                  .WORD   NOP             ;NOP INSTRUCTION
013366  000111          30024$: .WORD   111             ;JMP    (R1)
013370  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013372  000240                  .WORD   NOP             ;NOP INSTRUCTION
013374  000111          30025$: .WORD   111             ;JMP    (R1)
013376  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013400  000240                  .WORD   NOP             ;NOP INSTRUCTION
013402  000111          30026$: .WORD   111             ;JMP    (R1)
013404  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013406  000240                  .WORD   NOP             ;NOP INSTRUCTION
013410  000111          30027$: .WORD   111             ;JMP    (R1)
013412  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013414  000240                  .WORD   NOP             ;NOP INSTRUCTION
013416  000111          30028$: .WORD   111             ;JMP    (R1)
013420  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013422  000240                  .WORD   NOP             ;NOP INSTRUCTION
013424  000111          30029$: .WORD   111             ;JMP    (R1)
013426  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013430  000240                  .WORD   NOP             ;NOP INSTRUCTION
013432  000111          30030$: .WORD   111             ;JMP    (R1)
013434  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013436  000240                  .WORD   NOP             ;NOP INSTRUCTION
013440  000111          30031$: .WORD   111             ;JMP    (R1)
013442  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013444  000240                  .WORD   NOP             ;NOP INSTRUCTION
013446  000111          30032$: .WORD   111             ;JMP    (R1)
013450  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013452  000240                  .WORD   NOP             ;NOP INSTRUCTION
013454  000111          30033$: .WORD   111             ;JMP    (R1)
013456  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013460  000240                  .WORD   NOP             ;NOP INSTRUCTION
013462  000111          30034$: .WORD   111             ;JMP    (R1)
013464  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013466  000240                  .WORD   NOP             ;NOP INSTRUCTION
013470  000111          30035$: .WORD   111             ;JMP    (R1)
013472  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
013474  000240                  .WORD   NOP             ;NOP INSTRUCTION
013476  000111          30036$: .WORD   111             ;JMP    (R1)
013500  012717                  MOV     (PC)+,(PC)      ;WRITE THE NOP OVER THE JMP INSTRUCTION
```

```
********** DOUBLE OPERAND TESTS *****************

        013502  000240                          .WORD   NOP                     ;NOP INSTRUCTION
        013504  000111                  30037$: .WORD   111                     ;JMP    (R1)
        013506  012717                          MOV     (PC)+,(PC)      --      ;WRITE THE NOP OVER THE JMP INSTRUCTION
        013510  000240                          .WORD   NOP                     ;NOP INSTRUCTION
        013512  000111                  30038$: .WORD   111                     ;JMP    (R1)
        013514  012717                          MOV     (PC)+,(PC)              ;WRITE THE NOP OVER THE JMP INSTRUCTION
        013516  000240                          .WORD   NOP                     ;NOP INSTRUCTION
        013520  000111                  30039$: .WORD   111                     ;JMP    (R1)
  3531  013522  000137  013530                  JMP     @#129$                  ;JUMP OVER ERROR CALL
  3532  013526                        128$:                                   ;ERROR! PRE-FETCH BUFFER WAS NOT
  3533                                                                          ;OVER WRITTEN
  3534  013526  104001                          ERROR   +1              ;CPU ERROR
  3535                                  ;
  3536                                  ;       NOW RESTORE THE OVER WRITTEN JMP INSTRUCTIONS FOR THE NEXT PASS.
  3537                                  ;
  3538  013530  012702  000040         129$:   MOV     #32.,R2                 ;SET UP R2 AS COUNTER
  3539  013534  012703  013216                 MOV     #MJU7E,R3               ;SET UP R3 AS POINTER. $$$
  3540  013540  062703  000010                 ADD     #10,R3          ;ADD OFFSET TO POINT TO 1st. JMP IN TABLE. $$$
  3541  013544  012713  000111         130$:   MOV     #111,(R3)               ;RESTORE OVER WRITTEN JUMPS
  3542  013550  062703  000006                 ADD     #6,R3                   ;POINT TO NEXT OVER WRITTEN ADDR.
  3543  013554  077205                          SOB     R2,130$                 ;DO UNTIL R2=0
  3544
  3546                                  ;
  3548  013556                        MJP:
  3549
  3550                                  ;       TEST JMP MODES 17,27,37,67,77
  3551  013556  012737  000000  003072          MOV     #0,@#SEQ                ;SETUP TEST SEQUNCER
  3552  013564  000117                          JMP     (R7)                    ;JUMP MODE 17(SHOULD BE IN-LINE)
  3553                                  ;
  3554  013566  005737  003072         MJP17:  TST     @#SEQ                   ;CHECK SEQUENCE
  3555  013572  001401                          BEQ     2$                      ;BRANCH IF GOOD
  3556                                                                          ;ERROR! BAD JUMP
  3557  013574  104001                  1$:     ERROR   +1              ;CPU ERROR
  3558  013576  005237  003072         2$:     INC     @#SEQ                   ;UPDATE SEQUENCE NUMBER
Z 3559  013602  000127  000401                  JMP     #401                    ;JUMP MODE 27
  3560                                                                          ;(THE #401=BR UPDATED PC+2)
  3561                                  ;
  3562  013606  000000                          HALT                            ;
  3563                                  ;
  3564  013610  023727  003072  000001 MJP27:   CMP     @#SEQ,#1                ;CHECK IF CORRECT SEQUENCE
  3565  013616  001401                          BEQ     MJP27A                  ;BRANCH IF IN SEQUENCE
  3566  013620  104001                  1$:     ERROR   +1              ;CPU ERROR
  3567                                                                  ; TEST OUT OF SEQUENCE
  3568  013622  005237  003072         MJP27A: INC     @#SEQ                   ;UPDATE SEQUENCER
  3569  013626  000137  013674                  JMP     @#MJP37                 ;JUMP MODE 37
  3570  013632  023727  003072  000003 MJP67:   CMP     @#SEQ,#3                ;CHECK FOR CORRECT SEQUENCE
  3571  013640  001401                          BEQ     MJP67A                  ;BRANCH IF IN SEQUENCE
  3572  013642  104001                  2$:     ERROR   +1              ;CPU ERROR
  3573                                                                  ; TEST OUT OF SEQUENCE
  3574  013644  005237  003072         MJP67A: INC     @#SEQ                   ;UPDATE SEQUENCER
  3575  013650  000257                          CCC                             ;INSURE ZBIT=0
  3576  013652  000177  000002                  JMP     @MJP67B                 ;JUMP MODE 7
  3577                                  ;
  3578  013656  000000                          HALT                            ;
  3579                                  ;
  3580  013660  013662                 MJP67B: .WORD   MJP77                   ;
  3581                                  ;
```

```
********** DOUBLE OPERAND TESTS *****************

 3582 013662  023727  003072  000004  MJP77:  CMP     @#SEQ,#4          ; TEST FOR CORRECT SEQUENCE
 3583 013670  001412                          BEQ     MJP77E            ;BRANCH IF IN SEQUENCE
 3584 013672  104001                  3$:     ERROR   +1          ;CPU ERROR
 3585                                                              ; TEST OUT OF SEQUENCE
 3586 013674  023727  003072  000002  MJP37:  CMP     @#SEQ,#2          ; TEST FOR CORRECT SEQUENCE
 3587 013702  001401                          BEQ     MJP37A            ;BRANCH IF IN SEQUENCE
 3588 013704  104001                  4$:     ERROR   +1          ;CPU ERROR
 3589                                                              ; TEST OUT OF SEQUENCE
 3590 013706  005237  003072          MJP37A: INC     @#SEQ             ;UPDATE SEQUENCER
 3591 013712  000167  177714                  JMP     MJP67             ;JUMP MODE 6
 3592                                 ;
 3593                                 ;
 3594 013716                          MJP77E:
 3595                                 ;
 3596                                 ;
 3599 013716                          MJSR:
 3600
 3601                                 ;       TEST JSR ALL MODES
 3602 013716  010637  003074                  MOV     R6,@#SPS          ;SAVE STACK POINTER LOCATION
 3603 013722  010637  003076                  MOV     R6,@#SPSJ         ;
 3604 013726  162737  000002  003076          SUB     #2,@#SPSJ         ;SPSJ = R6 AFTER DECRIMENT
 3605 013734  012737  000001  003072          MOV     #1,@#SEQ          ;SETUP SEQUENCE COUNTER
 3606 013742  012701  014036                  MOV     #MJSR1,R1         ;SETUP INITIAL JUMP IN MODE 1
 3607 013746  005004                          CLR     R4                ;
 3608 013750  005104                          COM     R4                ;R4=-1 TO BE SAVED ON STACK
 3609 013752  004411                          JSR     R4,(R1)           ;JSR MODE 1
 3610                                 ;
 3611 013754  022737  000002  003072  MJSR2:  CMP     #2,@#SEQ          ; TEST FOR CORRECT SEQUENCE
 3612 013762  001401                          BEQ     MJSR2A            ;BRANCH IF GOOD
 3613 013764  104001                  5$:     ERROR   +1          ;CPU ERROR
 3614                                                              ;MODE 2 JUMPED TO OUT OF SEQUENCE
 3615 013766  023706  003076          MJSR2A: CMP     @#SPSJ,R6         ;VERIFY STACK DECRIMENT
 3616 013772  001006                          BNE     6$                ;BRANCH IF STACK INCORRECT
 3617 013774  021627  125252                  CMP     (R6),#125252      ;VERIFY CONTENTS OF STACK
 3618 014000  001003                          BNE     6$                ;BRANCH IF DATA ON STACK INCORRECT
 3619 014002  022704  014116                  CMP     #MJSR4,R4         ;SEE IF CORRECT RETURN ADDRESS
 3620 014006  001401                          BEQ     MJSR2B            ;BRANCH IF GOOD
 3621 014010  104001                  6$:     ERROR   +1          ;CPU ERROR
 3622                                                              ;JSR MODE 2 FAILED
 3623 014012  005237  003072          MJSR2B: INC     @#SEQ             ;UPDATE SEQUENCE COUNTER
 3624 014016  013706  003074                  MOV     @#SPS,R6          ;RELOAD STACK POINTER
 3625 014022  012701  014034                  MOV     #MJSRA,R1         ;SETUP JSR MODE 3
 3626 014026  005004                          CLR     R4                ;DIFFERENT DATA TO R4
 3627 014030  004431                          JSR     R4,@(R1)+         ;JSR MODE 3
 3628 014032  000000                          HALT
 3629 014034  014202                  MJSRA:  .WORD   MJSR3             ;LITERAL FOR JUMP MODE 3
 3630                                 ;
 3631 014036  022737  000001  003072  MJSR1:  CMP     #1,@#SEQ          ; TEST FOR CORRECT SEQUENCE
 3632 014044  001401                          BEQ     MJSR1A            ;BRANCH IF GOOD
 3633 014046  104001                  7$:     ERROR   +1          ;CPU ERROR
 3634                                                              ;MODE 1 JUMPED TO OUT OF SEQUENCE
 3635 014050  023706  003076          MJSR1A: CMP     @#SPSJ,R6         ;VERIFY STACK DECRIMENT
 3636 014054  001006                          BNE     8$                ;BRANCH IF STACK INCORRECT
 3637 014056  021627  177777                  CMP     (R6),#-1          ;VERIFY CONTENT OF STACK
 3638 014062  001003                          BNE     8$                ;BRANCH IF DATA ON STACK INCORRECT
 3639 014064  022704  013754                  CMP     #MJSR2,R4         ;SEE IF CORRECT RETURN ADDRESS
 3640 014070  001401                          BEQ     MJSR1B            ;BRANCH IF GOOD
```

```
********** DOUBLE OPERAND TESTS *****************

 3641 014072  104001                  8$:     ERROR   ;1              ;CPU ERROR
 3642                                                                  ;JSR MODE 2 FAILED
 3643 014074  005237  003072           MJSR1B: INC     @#SEQ                   ;UPDATE SEQUENCE COUNTER
 3644 014100  013706  003074                   MOV     @#SPS,R6                ;RELOAD STACK POINTER
 3645 014104  012704  125252                   MOV     #125252,R4              ;SETUP R4 DATA
 3646 014110  012701  013754                   MOV     #MJSR2,R1               ;SETUP MODE 2 JUMP ADDRESS
Z3647 014114  004421                           JSR     R4,(R1)+                ;*JUMP MODE 2
 3648                                  ;
 3649                                  ;
 3650 014116  022737  000004  003072   MJSR4:  CMP     #4,@#SEQ                ; TEST FOR CORRECT SEQUENCE
 3651 014124  001401                           BEQ     MJSR4A                  ;BRANCH IF GOOD
 3652 014126  104001                  9$:     ERROR   ;1              ;CPU ERROR
 3653                                                                  ;MODE 4 JUMPED TO OUT OF SEQUENCE
 3654 014130  023706  003076           MJSR4A: CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
 3655 014134  001006                           BNE     10$                     ;BRANCH IF STACK INCORRECT
 3656 014136  021627  052525                   CMP     (R6),#052525            ;VERIFY CONTENTS OF STAACK
 3657 014142  001003                           BNE     10$                     ;BRANCH IF DATA ON STACK INCORRECT
 3658 014144  022704  014264                   CMP     #MJSR6,R4               ;SEE IF CORRECT RETURN ADDRESS
 3659 014150  001401                           BEQ     MJSR4B                  ;BRANCH IF GOOD
 3660 014152  104001                  10$:    ERROR   ;1              ;CPU ERROR
 3661                                                                  ;JSR MODE 4 FAILED
 3662 014154  005237  003072           MJSR4B: INC     @#SEQ                   ;UPDATE SEQUENCE COUNTER
 3663 014160  013706  003074                   MOV     @#SPS,R6                ;RELOAD STACK POINTER
 3664 014164  012704  000377                   MOV     #377,R4                 ;SETUP R4 DATA
 3665 014170  012701  014202                   MOV     #MJSRB+2,R1             ;SETUP JSR VECTOR
 3666 014174  004451                           JSR     R4,@-(R1)               ;JSR MODE 5
 3667 014176  000000                           HALT
 3668 014200  014350                  MJSRB:  .WORD   MJSR5                   ;MODE 5 VECTOR
 3669                                  ;
 3670                                  ;
 3671 014202  022737  000003  003072   MJSR3:  CMP     #3,@#SEQ                ; TEST FOR CORRECT SEQUENCE
 3672 014210  001401                           BEQ     MJSR3A                  ;BRANCH IF GOOD
 3673 014212  104001                  11$:    ERROR   ;1              ;CPU ERROR
 3674                                                                  ;MODE 3 JUMPED TO OUT OF SEQUENCE
 3675 014214  023706  003076           MJSR3A: CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
 3676 014220  001006                           BNE     12$                     ;BRANCH IF STACK INCORRECT
 3677 014222  021627  000000                   CMP     (R6),#0                 ;VERIFY CONTENTS OF STAACK
 3678 014226  001003                           BNE     12$                     ;BRANCH IF DATA ON STACK INCORRECT
 3679 014230  022704  014032                   CMP     #MJSRA-2,R4             ;SEE IF CORRECT RETURN ADDRESS
 3680 014234  001401                           BEQ     MJSR3B                  ;BRANCH IF GOOD
 3681 014236  104001                  12$:    ERROR   ;1              ;CPU ERROR
 3682                                                                  ;JSR MODE 3 FAILED
 3683 014240  005237  003072           MJSR3B: INC     @#SEQ                   ;UPDATE SEQUENCE COUNTER
 3684 014244  013706  003074                   MOV     @#SPS,R6                ;RELOAD STACK POINTER
 3685 014250  012704  052525                   MOV     #052525,R4              ;SETUP R4 DATA
 3686 014254  012701  014120                   MOV     #MJSR4+2,R1             ;SETUP JSR VECTOR
 3687 014260  000257                           CCC                             ;CLEAR CONDITION CODES
 3688 014262  004441                           JSR     R4,-(R1)                        ;JSR MODE 4
 3689                                  ;
 3690                                  ;
 3691 014264  022737  000006  003072   MJSR6:  CMP     #6,@#SEQ                ; TEST FOR CORRECT SEQUENCE
 3692 014272  001401                           BEQ     MJSR6A                  ;BRANCH IF GOOD
 3693 014274  104001                  13$:    ERROR   ;1              ;CPU ERROR
 3694                                                                  ;MODE 6 JUMPED TO OUT OF SEQUENCE
 3695 014276  023706  003076           MJSR6A: CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
 3696 014302  001006                           BNE     14$                     ;BRANCH IF STACK INCORRECT
 3697 014304  021627  123456                   CMP     (R6),#123456            ;VERIFY CONTENTS OF STACK
```

```
********** DOUBLE OPERAND TESTS *****************

 3698 014310  001003                        BNE     14$                     ;BRANCH IF DATA ON STACK INCORRECT
 3699 014312  022704  014432                CMP     #MJSR7,R4               ;SEE IF CORRECT RETURN ADDRESS
 3700 014316  001401                        BEQ     MJSR6B                  ;BRANCH IF GOOD
 3701 014320  104001                14$:    ERROR   +1              ;CPU ERROR
 3702                                                               ;JSR MODE 6 FAILED
 3703 014322  005237  003072        MJSR6B: INC     @#SEQ                   ;UPDATE SEQUENCE COUNTER
 3704 014326  013706  003074                MOV     @#SPS,R6                ;RELOAD STACK POINTER
 3705 014332  012704  177773                MOV     #-5,R4                  ;SETUP R4 DATA
 3706 014336  012701  014356                MOV     #MJSRC+10,R1            ;SETUP JSR VECTOR
 3707 014342  004471  177770                JSR     R4;@-10(R1)             ;JSR MODE 7
 3708 014346  014432                MJSRC:  .WORD   MJSR7                   ;JSR VECTOR
 3709                                       ;
 3710                                       ;
 3711 014350  022737  000005  003072 MJSR5:  CMP     #5,@#SEQ                ; TEST FOR CORRECT SEQUENCE
 3712 014356  001401                        BEQ     MJSR5A                  ;BRANCH IF GOOD
 3713 014360  104001                15$:    ERROR   +1              ;CPU ERROR
 3714                                                               ;MODE 5 JUMPED TO OUT OF SEQUENCE
 3715 014362  023706  003076        MJSR5A: CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
 3716 014366  001006                        BNE     16$                     ;BRANCH IF STACK INCORRECT
 3717 014370  021627  000377                CMP     (R6),#377               ;VERIFY CONTENTS OF STACK
 3718 014374  001003                        BNE     16$                     ;BRANCH IF DATA ON STACK INCORRECT
 3719 014376  022704  014176                CMP     #MJSRB-2,R4             ;SEE IF CORRECT RETURN ADDRESS
 3720 014402  001401                        BEQ     MJSR5B                  ;BRANCH IF GOOD
 3721 014404  104001                16$:    ERROR   +1              ;CPU ERROR
 3722                                                               ;JSR MODE 5 FAILED
 3723 014406  005237  003072        MJSR5B: INC     @#SEQ                   ;UPDATE SEQUENCE COUNTER
 3724 014412  013706  003074                MOV     @#SPS,R6                ;RELOAD STACK POINTER
 3725 014416  012704  123456                MOV     #123456,R4              ;SETUP DATA IN R4
 3726 014422  012701  014274                MOV     #MJSR6+10,R1            ;SETUP JSR VECTOR
 3727 014426  004461  177770                JSR     R4,-10(R1)              ;JUMP MODE 6
 3728                                       ;
 3729                                       ;
 3730 014432  022737  000007  003072 MJSR7:  CMP     #7,@#SEQ                ; TEST FOR CORRECT SEQUENCE
 3731 014440  001401                        BEQ     MJSR7A                  ;BRANCH IF GOOD
 3732 014442  104001                17$:    ERROR   +1              ;CPU ERROR
 3733                                                               ;MODE 7 JUMPED TO OUT OF SEQUENCE
 3734 014444  023706  003076        MJSR7A: CMP     @#SPSJ,R6               ;VERIFY STACK DECRIMENT
 3735 014450  001006                        BNE     18$                     ;BRANCH IF STACK INCORRECT
 3736 014452  021627  177773                CMP     (R6),#-5                ;VERIFY CONTENTS OF STAACK
 3737 014456  001003                        BNE     18$                     ;BRANCH IF DATA ON STACK INCORRECT
 3738 014460  022704  014346                CMP     #MJSR5-2,R4             ;SEE IF CORRECT RETURN ADDRESS
 3739 014464  001401                        BEQ     MJSR7E                  ;BRANCH IF GOOD
 3740 014466  104001                18$:    ERROR   +1              ;CPU ERROR
 3741                                                               ;JSR MODE 7 FAILED
 3742 014470                        MJSR7E:
 3743 014470  013706  003074                MOV     @#SPS,R6                ;REPLACE STACK
 3744
 3745                                       ;
 3748 014474                        MJRA:
 3749
 3750                                       ;       TEST JSR MODES 27, 37, 67, 77
 3751 014474  012737  000001  003072        MOV     #1,@#SEQ                ;SETUP SEQUENCER
 3752 014502  010637  003074                MOV     R6,@#SPS                ;SAVE STACK ADDRESS
 3753 014506  010637  003076                MOV     R6,@#SPSJ               ;SAVE STACK DECRIMENT ADDRESS
 3754 014512  162737  000002  003076        SUB     #2,@#SPSJ               ;
 3755 014520  012704  177777                MOV     #-1,R4                  ;SETUP R4 DATA
Z 3756 014524  004427  000240                JSR     R4,#240                 ;EXECUTE A JSR MODE 27
```

```
********** DOUBLE OPERAND TESTS *****************

                                        ;
014530  022737  000001  003072  MJR27:  CMP     #1,@#SEQ        ;VERIFY COERRECT TEST SEQUENCE
014536  001011                          BNE     1$              ;INCORRECT TEST SEQUENCE
014540  023706  003076                  CMP     @#SPSJ,R6       ;VERIFY STACK POINTER
014544  001006                          BNE     1$
014546  021627  177777                  CMP     (R6),#-1        ;VERIFY R4 GOT LOADED ON THE STACK
014552  001003                          BNE     1$              ;BRANCH IF INCORRECT STACK CONTENTS
014554  020427  014530                  CMP     R4,#MJR27       ;VERIFY CORRECT RETURN ADDRESS
014560  001401                          BEQ     MJR27A          ;BRANCH IF GOOD RETURN ADDRESS ON STACK
014562  104001                  1$:     ERROR   +1      ;CPU ERROR
                                                        ;MODE 27 FAILED
014564  005237  003072          MJR27A: INC     @#SEQ           ;UPDATE SEQUENCER
014570  012704  152525                  MOV     #152525,R4      ;SETUP R4 TEST DATA
014574  013706  003074                  MOV     @#SPS,R6        ;RESET STACK POINTER
014600  004437  014666                  JSR     R4,@#MJR37      ;JSR MODE 37
014604  000000                  MJR27B: HALT
                                        ;
014606  023727  003072  000003  MJR67:  CMP     @#SEQ,#3        ;VERIFY TEST SEQUENCE
014614  001011                          BNE     2$              ;INCORRECT TEST SEQUENCE
014616  023706  003076                  CMP     @#SPSJ,R6       ;VERIFY STACK DECRIMENT
014622  001006                          BNE     2$              ;INCORRECT STACK DECRIMENT
014624  021627  000125                  CMP     (R6),#125       ;VERFIFY STACK WAS LOADED
014630  001003                          BNE     2$
014632  020427  014742                  CMP     R4,#MJR77       ;VERIFY RETURN ADDRESS
014636  001401                          BEQ     MJR67A          ;BRANCH IF GOOD
014640  104001                  2$:     ERROR   +1      ;CPU ERROR
                                                        ;MODE 67 FAILED
014642  005237  003072          MJR67A: INC     @#SEQ           ;UPDATE SEQUENCER
014646  013706  003074                  MOV     @#SPS,R6        ;RESET STACK
014652  012704  000001                  MOV     #1,R4           ;SETUP R4 DATA
014656  004477  000002                  JSR     R4,@MJR6B       ;JSR MODE 77
014662  000000                  MJR6A:  HALT
014664  014742                  MJR6B:  .WORD   MJR77           ;DATA FOR MODE 77 JUMP
                                        ;
014666  023727  003072  000002  MJR37:  CMP     @#SEQ,#2        ;VERIFY TEST SEQUENCE
014674  001011                          BNE     2$              ;INCORRECT TEST SEQUENCE
014676  023706  003076                  CMP     @#SPSJ,R6       ;VERIFY STACK DECRIMENT
014702  001006                          BNE     2$              ;INCORRECT STACK DECRIMENT
014704  021627  152525                  CMP     (R6),#152525    ;VERFIFY STACK WAS LOADED
014710  001003                          BNE     2$
014712  020427  014604                  CMP     R4,#MJR27B      ;VERIFY RETURN ADDRESS
014716  001401                          BEQ     MJR37A          ;BRANCH IF GOOD
014720  104001                  2$:     ERROR   +1      ;CPU ERROR
                                                        ;MODE 37 FAILED
014722  005237  003072          MJR37A: INC     @#SEQ           ;UPDATE SEQUENCER
014726  013706  003074                  MOV     @#SPS,R6        ;RELOAD STACK
014732  012704  000125                  MOV     #125,R4         ;SETUP R4 TEST DATA
014736  004467  177644                  JSR     R4,MJR67        ;JSR MODE 6
                                        ;
014742  023727  003072  000004  MJR77:  CMP     @#SEQ,#4        ;VERIFY TEST SEQUENCE
014750  001011                          BNE     2$              ;INCORRECT TEST SEQUENCE
014752  023706  003076                  CMP     @#SPSJ,R6       ;VERIFY STACK DECRIMENT
014756  001006                          BNE     2$              ;INCORRECT STACK DECRIMENT
014760  021627  000001                  CMP     (R6),#1         ;VERFIFY STACK WAS LOADED
014764  001003                          BNE     2$
014766  020427  014662                  CMP     R4,#MJR6A       ;VERIFY RETURN ADDRESS
014772  001401                          BEQ     MJR77A          ;BRANCH IF GOOD
```

```
********** DOUBLE OPERAND TESTS *****************

3814 014774  104001                  2$:     ERROR   +1              ;CPU ERROR
3815                                                                 ;MODE 77 FAILED
3816 014776                           MJR77A:
3817                                  ;
3818 014776  013706  003074                   MOV     @#SPS,R6                ;RESET STACK
3819
3820                                  ;
3823 015002                           MRTS:
3824
3825                                  ;       TEST RTS AND RTS R6
3826 015002  012706  001000                   MOV     #STBOT,R6               ;INSURE VALID STACK
3827 015006  012746  123456                   MOV     #123456,-(R6)           ;SETUP TEST REGISTER
3828 015012  012703  015022                   MOV     #RTS1,R3                ;SETUP TEST PC
3829 015016  000203                           RTS     R3                      ;**TEST INSTRUCTION
3830 015020  104001                           ERROR   +1              ;CPU ERROR
3831                                                                 ;INCORRECT PC ON RTS
3832 015022  022703  123456           RTS1:   CMP     #123456,R3
3833 015026  001401                           BEQ     RTS6                    ;BRANCH IF GOOD
3834 015030  104001                           ERROR   +1              ;CPU ERROR
3835                                                                 ;REGISTER CONTENTS INCORRECT
3836                                         ;
3837                                         ;THIS TEST CHECKS AN UN-TESTED PLA TERM
3838                                         ;
3839 015032  010601                   RTS6:   MOV     R6,R1                   ;SAVE STACK IN R1
3840 015034  012705  015046                   MOV     #1$,R5                  ;MOVE EXPECTED RETURN ADDR TO R5
3841 015040  010506                           MOV     R5,R6                   ;MOVE RETURN ADDR TO R6
3842 015042  000206                           RTS     R6                      ;
3843 015044  104001                           ERROR   +1              ;CPU ERROR
3844                                                                 ;ERROR! RTS NOT EXECUTED
3845 015046  021506                   1$:     CMP     (R5),R6                 ;IS R6=31506?
3846 015050  001401                           BEQ     RTSE                    ;IF IT IS THEN GO TO END OF TEST
3847 015052  104001                           ERROR   +1              ;CPU ERROR
3848                                                                 ;ERROR! WRONG ADDR IN R6
3849 015054  010106                   RTSE:   MOV     R1,R6                   ;RESTORE STACK
3850
3853 015056                           TSMMUO:
3854                                  ;       SETUP AND TEST KERNEL, SUPERVISOR AND USER STACKS
3855 015056  012737  040000  177776           MOV     #40000,@#177776         ;SET PS TO SUP MODE
3856 015064  012706  177777                   MOV     #177777,R6              ;INIT SUP STACK TO ALL ONES
3857 015070  022706  177777                   CMP     #177777,R6              ;ARE ALL BITS SET
3858 015074  001401                           BEQ     1$                      ;YES GO ON
3859                                                                         ;NO GO TO ERROR
3860 015076  104001                           ERROR   +1              ;CPU ERROR
3861 015100  005006                   1$:     CLR     R6                      ;SET SUP STACK TO ALL ZEROES
3862 015102  022706  000000                   CMP     #0,R6                   ;ARE ALL BITS CLEARED
3863 015106  001401                           BEQ     2$                      ;YES GO ON
3864                                                                         ;NO GO TO ERROR
3865 015110  104001                           ERROR   +1              ;CPU ERROR
3866 015112  012706  125252           2$:     MOV     #125252,R6              ;SET SUP STACK TO ALTERNATING PATTERN
3867 015116  022706  125252                   CMP     #125252,R6              ;IS SUP SP CORRECT
3868 015122  001401                           BEQ     3$                      ;YES GO ON
3869                                                                         ;NO GO TO ERROR
3870 015124  104001                           ERROR   +1              ;CPU ERROR
3871 015126  012706  052525           3$:     MOV     #52525,R6               ;SET SUP STACK TO ALTERNATING PATTERN
3872 015132  022706  052525                   CMP     #52525,R6               ;IS SUP SP CORRECT
3873 015136  001401                           BEQ     4$                      ;YES GO ON
3874                                                                         ;NO GO TO ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

3875 015140  104001                        ERROR   ;1              ;CPU ERROR
3876 015142  012706  000700          4$:    MOV     #700,R6                 ;SETUP SUP SP
3877 015146  012737  140000  177776        MOV     #140000,@#177776        ;SET PS TO USER MODE
3878 015154  012706  177777                MOV     #177777,R6              ;INIT USER STACK TO ALL ONES
3879 015160  022706  177777                CMP     #177777,R6              ;ARE ALL BITS SET
3880 015164  001401                        BEQ     5$                      ;YES GO ON
3881                                                                       ;NO GO TO ERROR
3882 015166  104001                        ERROR   ;1              ;CPU ERROR
3883 015170  005006                  5$:   CLR     R6                      ;SET USER STACK TO ALL ZEROES
3884 015172  022706  000000                CMP     #0,R6                   ;ARE ALL BITS CLEARED
3885 015176  001401                        BEQ     6$                      ;YES GO ON
3886                                                                       ;NO GO TO ERROR
3887 015200  104001                        ERROR   ;1              ;CPU ERROR
3888 015202  012706  125252          6$:   MOV     #125252,R6              ;SET USER STACK TO ALTERNATING PATTERN
3889 015206  022706  125252                CMP     #125252,R6              ;IS USER SP CORRECT
3890 015212  001401                        BEQ     7$                      ;YES GO ON
3891                                                                       ;NO GO TO ERROR
3892 015214  104001                        ERROR   ;1              ;CPU ERROR
3893 015216  012706  052525          7$:   MOV     #52525,R6               ;SET USER STACK TO ALTERNATING PATTERN
3894 015222  022706  052525                CMP     #52525,R6               ;IS USER SP CORRECT
3895 015226  001401                        BEQ     8$                      ;YES GO ON
3896                                                                       ;NO GO TO ERROR
3897 015230  104001                        ERROR   ;1              ;CPU ERROR
3898 015232  012706  000600          8$:   MOV     #600,R6                 ;SETUP USER SP
3899 015236  005037  177776                CLR     @#177776                ;SET PS TO KER MODE
3900 015242  012706  001000                MOV     #STBOT,R6               ;SETUP KERNEL SP
3901 015246  005037  000700                CLR     @#700                   ;CLEAR FIRST WORDS OF SUP, KER, AND USE STACKS
3902 015252  005037  000600                CLR     @#600                   ;
3903 015256  005037  001000                CLR     @#STBOT                 ;
3904 015262  004767  000054                JSR     PC,CHECK                ; TEST KER, SUP, AND USE STACKS
3905 015266  012737  040000  177776  RET1: MOV     #40000,@#177776         ;SET PSW TO SUP MODE
3906 015274  022706  000700                CMP     #700,R6                 ;IS SUP SP CORRECT
3907 015300  001401                        BEQ     1$                      ;YES GO ON
3908                                                                       ;NO GO TO ERROR
3909 015302  104001                        ERROR   ;1              ;CPU ERROR
3910 015304  012737  140000  177776  1$:   MOV     #140000,@#177776        ;SET PSW TO USE MODE
3911 015312  022706  000600                CMP     #600,R6                 ;IS USE SP CORRECT
3912 015316  001401                        BEQ     2$                      ;YES GO ON
3913                                                                       ;NO GO TO ERROR
3914 015320  104001                        ERROR   ;1              ;CPU ERROR
3915 015322  005037  177776          2$:   CLR     @#177776                ;SET PSW TO KER MODE
3916 015326  022706  001000                CMP     #STBOT,R6               ;IS KER SP CORRECT
3917 015332  001401                        BEQ     3$                      ;YES GO ON
3918                                                                       ;NO GO TO ERROR
3919 015334  104001                        ERROR   ;1              ;CPU ERROR
3920 015336                          3$:
3921 015336  000167  000206                JMP     MTSO
3922                                ;
3923                                ;ROUTINE TO CHECK STACKS AFTER TWO RTS STATEMENTS
3924                                ;
3925 015342  012737  040000  177776  CHECK: MOV     #40000,@#177776         ;SET PSW TO SUP MODE
3926 015350  004767  000044                JSR     PC,CHECK1               ; TEST SUP, KER, AND USE STACKS
3927 015354  022716  000000          RET2: CMP     #0,(SP)                 ;IS SUP STACK CLEARED
3928 015360  001401                        BEQ     1$                      ;YES GO ON
3929                                                                       ;NO GO TO ERROR
3930 015362  104001                        ERROR   ;1              ;CPU ERROR
3931 015364  012737  140000  177776  1$:   MOV     #140000,@#177776        ;SET PSW TO USE MODE
```

```
********** DOUBLE OPERAND TESTS *****************

3932 015372  022716  000000                  CMP     #0,(SP)                 ;IS USE STACK CLEARED
3933 015376  001401                          BEQ     2$                      ;YES GO ON
3934                                                                         ;NO GO TO ERROR
3935 015400  104001                          ERROR   +1              ;CPU ERROR
3936 015402  005037  177776          2$:     CLR     @#177776                ;SET PSW TO KER MODE
3937 015406  022716  015266                  CMP     #RET1,(SP)              ;DOES KER STACK HAVE CORRECT DATA
3938 015412  001401                          BEQ     3$                      ;YES GO ON
3939                                                                         ;NO GO TO ERROR
3940 015414  104001                          ERROR   +1              ;CPU ERROR
3941 015416  000207                  3$:     RTS     PC                      ;RETURN
3942                                 ;
3943                                 ;ROUTINE TO CHECK STACKS AFTER ONE RTS
3944                                 ;
3945 015420  012737  140000  177776  CHECK1: MOV     #140000,@#177776        ;SET PSW TO USE MODE
3946 015426  004767  000044                  JSR     PC,CHECK2               ; TEST KER, SUP, AND USE STACKS
3947 015432  022716  000000          RET3:   CMP     #0,(SP)                 ;IS USE STACK CLEARED
3948 015436  001401                          BEQ     1$                      ;YES GO ON
3949                                                                         ;NO GO TO ERROR
3950 015440  104001                          ERROR   +1              ;CPU ERROR
3951 015442  005037  177776          1$:     CLR     @#177776                ;SET PSW TO KER MODE
3952 015446  022716  015266                  CMP     #RET1,(SP)              ;IS KER STACK CORRECT
3953 015452  001401                          BEQ     2$                      ;YES GO ON
3954                                                                         ;NO GO TO ERROR
3955 015454  104001                          ERROR   +1              ;CPU ERROR
3956 015456  012737  040000  177776  2$:     MOV     #40000,@#177776         ;SET PSW TO SUP MODE
3957 015464  022716  015354                  CMP     #RET2,(SP)              ;IS SUP STACK CORRECT
3958 015470  001401                          BEQ     3$                      ;YES GO ON
3959                                                                         ;NO GO TO ERROR
3960 015472  104001                          ERROR   +1              ;CPU ERROR
3961 015474  000207                  3$:     RTS     PC                      ;RETURN
3962                                 ;
3963                                 ;ROUTINE TO CHECK STACKS AFTER ZERO RTS
3964                                 ;
3965 015476  022716  015432          CHECK2: CMP     #RET3,(SP)              ;IS USE STACK CORRECT
3966 015502  001401                          BEQ     1$                      ;YES GO ON
3967                                                                         ;NO GO TO ERROR
3968 015504  104001                          ERROR   +1              ;CPU ERROR
3969 015506  012737  040000  177776  1$:     MOV     #40000,@#177776         ;SET PSW TO SUP MODE
3970 015514  022716  015354                  CMP     #RET2,(SP)              ;IS SUP STACK CORRECT
3971 015520  001401                          BEQ     2$                      ;YES GO ON
3972                                                                         ;NO GO TO ERROR
3973 015522  104001                          ERROR   +1              ;CPU ERROR
3974 015524  005037  177776          2$:     CLR     @#177776                ;SET PSW TO KER MODE
3975 015530  022716  015266                  CMP     #RET1,(SP)              ;IS KER STACK CORRECT
3976 015534  001401                          BEQ     3$                      ;YES GO ON
3977                                                                         ;NO GO TO ERROR
3978 015536  104001                          ERROR   +1              ;CPU ERROR
3979 015540  012737  140000  177776  3$:     MOV     #140000,@#177776        ;SET PSW TO USE MODE
3980 015546  000207                          RTS     PC                      ;RETURN
3981                                 ;
3982 015550                          MTSO:
3985 015550                          MMVCC:
3986
3987                                 ;       TEST MOV CONDITION CODES - **0-
3988 015550  000277                          SCC                             ;
3989 015552  000244                          CLZ                             ;CC=1011
3990 015554  012704  000000                  MOV     #0,R4                   ;CC=0101, R4=0
```

```
********** DOUBLE OPERAND TESTS ******************

3991 015560  100403                        BMI     1$                      ;ERROR IF N FLAG
3992 015562  102402                        BVS     1$                      ;ERROR IF V FLAG SET
3993 015564  103001                        BCC     1$                      ;ERROR IF C FLAG CLEAR
3994 015566  001401                        BEQ     2$                      ;SKIP IF Z FLAG SET
3995                                                                       ;CC SHOULD=0101
3996 015570  104001                 1$:    ERROR   +1              ;CPU ERROR
3997 015572  000277                 2$:    SCC
3998 015574  000251                        .WORD   251                     ;CC=0110
3999 015576  012704  100000                MOV     #100000,R4              ;R4=100000, CC=1000
4000 015602  001403                        BEQ     3$                      ;ERROR IF Z SET
4001 015604  102402                        BVS     3$                      ;ERROR IF V SET
4002 015606  103401                        BCS     3$                      ;ERROR IF C SET
4003 015610  100401                        BMI     4$                      ;EXIT IF N SET
4004 015612  104001                 3$:    ERROR   +1              ;CPU ERROR
4005                                                               ;CC SHOULD= 1000
4006 015614                         4$:
4007
4008
4011 015614                         MBTCC:
4012
4013                                ;      TEST BIT CONDITION CODES - **0-
4014 015614  005004                        CLR     R4
4015 015616  005104                        COM     R4                      ;R4=-1
4016 015620  000277                        SCC                             ;
4017 015622  000244                        CLZ                             ;CC=1011
4018 015624  032704  000000                BIT     #0,R4                   ;CC=0101
4019 015630  100403                        BMI     1$                      ;ERROR IF N FLAG
4020 015632  102402                        BVS     1$                      ;ERROR IF V FLAG SET
4021 015634  103001                        BCC     1$                      ;ERROR IF C FLAG CLEAR
4022 015636  001401                        BEQ     2$                      ;SKIP IF Z FLAG SET
4023 015640  104001                 1$:    ERROR   +1              ;CPU ERROR
4024                                                               ;CC SHOULD=0101
4025 015642  000277                 2$:    SCC
4026 015644  000251                        .WORD   251                     ;CC=0110
4027 015646  032704  100000                BIT     #100000,R4              ;CC=1000
4028 015652  001403                        BEQ     3$                      ;ERROR IF Z SET
4029 015654  102402                        BVS     3$                      ;ERROR IF V SET
4030 015656  103401                        BCS     3$                      ;ERROR IF C SET
4031 015660  100401                        BMI     4$                      ;EXIT IF N SET
4032 015662  104001                 3$:    ERROR   +1              ;CPU ERROR
4033                                                               ;CC SHOULD= 1000
4034 015664                         4$:
4035
4036
4039 015664                         MBCCC:
4040
4041                                ;      TEST BIC CONDITION CODES - **0-
4042 015664  005004                        CLR     R4
4043 015666  005104                        COM     R4                      ;R4=-1
4044 015670  000277                        SCC                             ;
4045 015672  000244                        CLZ                             ;CC=1011
4046 015674  042704  177777                BIC     #177777,R4              ;CC=0101
4047 015700  100403                        BMI     1$                      ;ERROR IF N FLAG
4048 015702  102402                        BVS     1$                      ;ERROR IF V FLAG SET
4049 015704  103001                        BCC     1$                      ;ERROR IF C FLAG CLEAR
4050 015706  001401                        BEQ     2$                      ;SKIP IF Z FLAG SET
4051 015710  104001                 1$:    ERROR   +1              ;CPU ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

4052
4053 015712  005104                  2$:     COM     R4                      ;CC SHOULD=0101
4054 015714  000277                          SCC                                     ;R4=-1
4055 015716  000251                          .WORD   251                             ;CC=0110
4056 015720  042704  077777                  BIC     #77777,R4               ;CC=1000
4057 015724  001403                          BEQ     3$                      ;ERROR IF Z SET
4058 015726  102402                          BVS     3$                      ;ERROR IF V SET
4059 015730  103401                          BCS     3$                      ;ERROR IF C SET
4060 015732  100401                          BMI     4$                      ;EXIT IF N SET
4061 015734  104001                  3$:     ERROR   .+1                     ;CPU ERROR
4062                                                                          ;CC SHOULD= 1000
4063 015736                          4$:
4064
4065
4068 015736                          MBSCC:
4069
4070                                 ;       TEST BIS CONDITION CODES
4071 015736  005004                          CLR     R4                              ;R4=0
4072 015740  000277                          SCC
4073 015742  000246                          .WORD   246                             ;CC=1001
4074 015744  052704  000000                  BIS     #0,R4                   ;R4=0, CC=0101
4075 015750  100403                          BMI     1$                      ;ERROR IF MINUS
4076 015752  102402                          BVS     1$                      ;ERROR IF V SET
4077 015754  103001                          BCC     1$                      ;ERROR IF C CLEAR
4078 015756  001401                          BEQ     2$                      ;BRANCH IF GOOD
4079 015760  104001                  1$:     ERROR   .+1                     ;CPU ERROR
4080                                                                          ;BIS CC FAILED
4081 015762  000277                  2$:     SCC
4082 015764  000241                          CLC                                     ;CC=1110
4083 015766  052704  100076                  BIS     #100076,R4              ;R4=100076, CC=1000
4084 015772  001403                          BEQ     3$                      ;ERROR IF Z SET
4085 015774  102402                          BVS     3$                      ;ERROR IF V SET
4086 015776  103401                          BCS     3$                      ;ERROR IF C SET
4087 016000  100401                          BMI     4$                      ;BRANCH IF GOOD
4088 016002  104001                  3$:     ERROR   .+1                     ;CPU ERROR
4089                                                                          ;BAD BIS CC
4090 016004                          4$:
4091
4092
4095 016004                          MDCCC:
4096
4097                                 ;       TEST DEC, INC CONDITION CODES
4098 016004  012704  077777                  MOV     #77777,R4                       ;R4=77777
4099 016010  000257                          CCC
4100 016012  000261                          SEC                                     ;CC=0001
4101 016014  005204                          INC     R4                      ;R4=100000, CC=0011
4102 016016  001403                          BEQ     1$                      ;ERROR IF ZERO
4103 016020  100002                          BPL     1$                      ;ERROR IF POSITIVE
4104 016022  102001                          BVC     1$                      ;ERROR IF V CLEAR
4105 016024  103401                          BCS     2$                      ;BRANCH IF GOOD
4106 016026  104001                  1$:     ERROR   .+1                     ;CPU ERROR
4107                                                                          ;INC FAILED
4108 016030  000257                  2$:     CCC
4109 016032  005204                          INC     R4                              ;R4=100001, CC=1000
4110 016034  103413                          BCS     3$                      ;ERROR IF C SET
4111 016036  102412                          BVS     3$                      ;ERROR IF V SET
4112 016040  005304                          DEC     R4                      ;R4=100000, CC=1000
```

```
********** DOUBLE OPERAND TESTS *****************

4113 016042  102410                    BVS     3$                      ;ERROR IF V SET
4114 016044  103407                    BCS     3$                      ;ERROR IF C SET
4115 016046  000277                    SCC                             ;
4116 016050  000252                    .WORD   252                     ;CC=0101
4117 016052  005304                    DEC     R4                      ;R4=77777, CC=1011
4118 016054  001403                    BEQ     3$                      ;ERROR IF Z SET
4119 016056  102002                    BVC     3$                      ;ERROR IF V CLEAR
4120 016060  103001                    BCC     3$                      ;ERROR IF C CLEAR
4121 016062  100001                    BPL     4$                      ;BRANCH IF GOOD
4122 016064  104001             3$:    ERROR   +1              ;CPU ERROR
4123                                                            ;BAD CC
4124 016066                      4$:
4125
4126
4136 016066                      MCTSCC:
4137
4138                             ;       TEST CLR, TST, SWAB CONDITION CODES
4139                             ;;*****************************************************************
                                 ;0100 - **00 - **00
4140 016066  000277                    SCC
4141 016070  000244                    CLZ                             ;CC=1011
4142 016072  005004                    CLR     R4                      ;R4=0, CC=0100
4143 016074  100403                    BMI     1$                      ;ERROR IF MINUS
4144 016076  102402                    BVS     1$                              ;ERROR IF V SET
4145 016100  103401                    BCS     1$                      ;ERROR IF C SET
4146 016102  001401                    BEQ     2$                      ;BRANCH IF GOOD
4147 016104  104001             1$:    ERROR   +1              ;CPU ERROR
4148                                                            ;BAD CC ON CLR
4149 016106  005104             2$:    COM     R4                      ;R4=-1
4150 016110  000277                    SCC                             ;CC=1111
4151 016112  005704                    TST     R4                      ;CC=1000
4152 016114  001403                    BEQ     3$                      ;ERROR IF Z SET
4153 016116  102402                    BVS     3$                      ;ERROR IF V SET
4154 016120  103401                    BCS     3$                      ;ERROR IF C SET
4155 016122  100401                    BMI     4$                      ;BRANCH IF GOOD
4156 016124  104001             3$:    ERROR   +1              ;CPU ERROR
4157                                                            ;BAD TST CC
4158 016126  000277             4$:    SCC                             ;CC=1111
4159 016130  000304                    SWAB    R4                      ;CC=1000
4160 016132  102402                    BVS     5$                      ;ERROR IF V SET
4161 016134  103401                    BCS     5$                      ;ERROR IF C SET
4162 016136  100401                    BMI     6$                      ;BRANCH IF GOOD
4163 016140  104001             5$:    ERROR   +1              ;CPU ERROR
4164                                                            ;BAD SWAB CC
4165 016142                      6$:
4166
4167
4170 016142                      MADCC:
4171
4172                             ;       TEST ADD CONDITION CODES
4173 016142  012704  077777               MOV     #77777,R4               ;R4=77777
4174 016146  012701  000001               MOV     #1,R1                   ;R1=1
4175 016152  000257                       CCC                             ;CC=0000
4176 016154  060401                       ADD     R4,R1                   ;77777 + 1 = 100000 IN R1
4177 016156  102003                       BVC     1$                      ;ERROR IF V CLEAR
4178 016160  103402                       BCS     1$                      ;ERROR IF CARRY
4179 016162  001401                       BEQ     1$                      ;ERROR IF Z SET
```

```
********** DOUBLE OPERAND TESTS *****************

 4180 016164  100401                    BMI     2$              ;BRANCH IF GOOD
 4181 016166  104001          1$:       ERROR   +1              ;CPU ERROR
 4182                                                           ;CC SHOULD =1010
 4183 016170  005204          2$:       INC     R4              ;R4=100000
 4184 016172  060401                    ADD     R4,R1           ;100000 + 100000 = 0 IN R1
 4185 016174  102002                    BVC     3$              ;ERROR IF V CLEAR
 4186 016176  103001                    BCC     3$              ;ERROR IF CARRY  CLEAR
 4187 016200  001401                    BEQ     4$              ;BRANCH IF GOOD
 4188 016202  104001          3$:       ERROR   +1              ;CPU ERROR
 4189                                                           ;CC SHOULD = 0111
 4190 016204  060401          4$:       ADD     R4,R1           ;0 + 100000 = 100000
 4191 016206  102402                    BVS     5$              ;ERROR IF V SET
 4192 016210  103401                    BCS     5$              ;ERROR IF  SET
 4193 016212  100401                    BMI     6$              ;BRANCH IF GOOD
 4194 016214  104001          5$:       ERROR   +1              ;CPU ERROR
 4195                                                           ;CC SHOULD = 1000
 4196 016216                  6$:
 4197
 4198
 4201 016216                  MACCC:
 4202
 4203                         ;         TEST ADC CONDITION CODES
 4204 016216  012704  177777            MOV     #177777,R4      ;R4=177777
 4205 016222  000277                    SCC                     ;CC=1111
 4206 016224  005504                    ADC     R4              ;R4=0 CC=0101
 4207 016226  100403                    BMI     1$              ;ERROR IF MINUS
 4208 016230  102402                    BVS     1$              ;ERROR IF V SET
 4209 016232  103001                    BCC     1$              ;ERROR IF C SET
 4210 016234  001401                    BEQ     2$              ;BRANCH IF GOOD
 4211 016236  104001          1$:       ERROR   +1              ;CPU ERROR
 4212                                                           ;BAD ADC
 4213 016240  012704  077777  2$:       MOV     #077777,R4      ;R4=077777
 4214 016244  000277                    SCC                     ;
 4215 016246  000242                    CLV                     ;CC=1101
 4216 016250  005504                    ADC     R4              ;R4=100000 CC=1010
 4217 016252  100003                    BPL     3$              ;ERROR IF PLUS
 4218 016254  103402                    BCS     3$              ;ERROR IF C SET
 4219 016256  001401                    BEQ     3$              ;ERROR IF ZERO
 4220 016260  102401                    BVS     4$              ;BRANCH IF GOOD
 4221 016262  104001          3$:       ERROR   +1              ;CPU ERROR
 4222                                                           ;BAD ADC
 4223 016264  000277          4$:       SCC                     ;CC=1111
 4224 016266  005504                    ADC     R4              ;R4=100000 CC=1000
 4225 016270  102402                    BVS     5$              ;ERROR IF V SET
 4226 016272  103401                    BCS     5$              ;ERROR IF C SET
 4227 016274  100401                    BMI     6$              ;BRANCH IF GOOD
 4228 016276  104001          5$:       ERROR   +1              ;CPU ERROR
 4229                                                           ;BAD ADC CC SHOULD= 1000
 4230 016300                  6$:
 4231
 4232
 4235 016300                  MNCCCC:
 4236
 4237                         ;         TEST NEG, CMP, COM CONDITION CODES
 4238 016300  012704  077777            MOV     #077777,R4      ;R4=77777
 4239 016304  000257                    CCC                     ;CC=0000
 4240 016306  005404                    NEG     R4              ;R4=100001      CC=1001
```

```
********** DOUBLE OPERAND TESTS *****************

4241 016310  102403                      BVS     1$              ;ERROR IF V SET
4242 016312  103002                      BCC     1$              ;ERROR IF C CLEAR
4243 016314  001401                      BEQ     1$              ;ERROR IF Z SET
4244 016316  100401                      BMI     2$              ;BRANCH IF GOOD
4245 016320  104001              1$:     ERROR   .+1             ;CPU ERROR
4246                                                             ;BAD NEGATE
4247 016322  005004              2$:     CLR     R4              ;R4=0
4248 016324  000257                      CCC                     ;CC=0000
4249 016326  005404                      NEG     R4              ;CC=0101
4250 016330  100403                      BMI     3$              ;ERROR IF N SET
4251 016332  103402                      BCS     3$              ;ERROR IF C SET
4252 016334  102401                      BVS     3$              ;ERROR IF V SET
4253 016336  001401                      BEQ     4$              ;BRANCH IF GOOD
4254 016340  104001              3$:     ERROR   .+1             ;CPU ERROR
4255                                                             ;BAD NEG
4256 016342  012704  077777      4$:     MOV     #77777,R4       ;R4=77777
4257 016346  012701  170000              MOV     #170000,R1      ;R1=170000
4258 016352  000257                      CCC                     ;CC=0000
4259 016354  020401                      CMP     R4,R1           ;77777 - 170000 = 107777 CC= 1011
4260 016356  102003                      BVC     5$              ;ERROR IF V CLEAR
4261 016360  103002                      BCC     5$              ;ERROR IF C CLEAR
4262 016362  001401                      BEQ     5$              ;ERROR IF ZERO
4263 016364  100401                      BMI     6$              ;BRANCH IF GOOD
4264 016366  104001              5$:     ERROR   .+1             ;CPU ERROR
4265                                                             ;BAD CMP
4266 016370  000257              6$:     CCC
4267 016372  005101                      COM     R1              ;R1=7777
4268 016374  100403                      BMI     7$              ;ERROR IF MINUS
4269 016376  001402                      BEQ     7$              ;ERROR IF ZERO
4270 016400  103001                      BCC     7$              ;ERROR IF CARRY
4271 016402  102001                      BVC     8$              ;BRANCH IF GOOD
4272 016404  104001              7$:     ERROR   .+1             ;CPU ERROR
4273                                                             ;BAD COM
4274 016406  000277              8$:     SCC
4275 016410  005101                      COM     R1
4276 016412  100401                      BMI     10$             ;BRANCH IF GOOD
4277 016414  104001              9$:     ERROR   .+1             ;CPU ERROR
4278                                                             ;BAD COM
4279 016416                      10$:
4280
4281
4284 016416                      MSBCC:
4285
4286                             ;       TEST SUB CONDITION CODES
4287 016416  012704  077775              MOV     #77775,R4       ;R4=77775
4288 016422  000257                      CCC                     ;CC=0000
4289 016424  162704  137757              SUB     #137757,R4      ;77775 - 137757
4290                                                             ;TRY TO CAUSE AN ARITHMETIC OVERFLOW
4291 016430  102003                      BVC     1$              ;ERROR IF V CLEAR
4292 016432  100002                      BPL     1$              ;ERROR IF RESULT IS POSITIVE.
4293 016434  001401                      BEQ     1$              ;ERROR IF  Z SET
4294 016436  103401                      BCS     2$              ;BRANCH IF GOOD
4295 016440  104001              1$:     ERROR   .+1             ;CPU ERROR
4296                                                             ;BAD SUBTRACT
4297 016442  012704  000005      2$:     MOV     #5,R4           ;R4=5
4298 016446  000257                      CCC                     ;CC=0000
4299 016450  162704  000012              SUB     #12,R4          ;5-12=-5 AND SETS CARRY
```

```
********** DOUBLE OPERAND TESTS *****************

 4300 016454  103003                       BCC     3$                      ;ERROR IF CARRY CLEAR
 4301 016456  102402                       BVS     3$                      ;ERROR IF OVERFLOW
 4302 016460  001401                       BEQ     3$                      ;ERROR IF ZERO
 4303 016462  100401                       BMI     4$                      ;BRANCH IF GOOD
 4304 016464  104001             3$:       ERROR   +1              ;CPU ERROR
 4305                                                               ;SUBTRACT FAILED
 4306 016466                     4$:
 4307
 4308
 4311 016466                     MSBCCC:
 4312
 4313                            ;       TEST SBC CONDITION CODES
 4314 016466  012704  100000               MOV     #100000,R4              ;R4=100000
 4315 016472  000257                       CCC                             ;C=0000
 4316 016474  005604                       SBC     R4                      ;TRY TO SET V
 4317 016476  100006                       BPL     1$                      ;ERROR IF N CLEAR
 4318 016500  102405                       BVS     1$                      ;ERROR IF V SET (HAVENT SET C YET)
 4319 016502  000261                       SEC                             ;CC SHOULD = 1001
 4320 016504  005604                       SBC     R4                      ;TRY AGAIN TO SET V
 4321 016506  102002                       BVC     1$                      ;ERROR IF V CLEAR
 4322 016510  103401                       BCS     1$                      ;ERROR IF C SET
 4323 016512  100001                       BPL     2$                      ;BRANCH IF GOOD
 4324 016514  104001             1$:       ERROR   +1              ;CPU ERROR
 4325                                                               ;SBC FAILED
 4326 016516  005004             2$:       CLR     R4                      ;R4=0
 4327 016520  000277                       SCC
 4328 016522  000241                       CLC                             ;CC=1110
 4329 016524  005604                       SBC     R4                      ;TRY TO CAUSE C FLAG FAILURE
 4330 016526  103410                       BCS     3$                      ;ERROR IF C SET
 4331 016530  102407                       BVS     3$                      ;ERROR IF V SET
 4332 016532  001006                       BNE     3$                      ;ERROR IF  NOT ZERO
 4333 016534  000261                       SEC                             ;SET CARRY
 4334 016536  005604                       SBC     R4                      ;NOW, 0 - CARRY = 177777
 4335 016540  103003                       BCC     3$                      ;ERROR IF CARRY CLEAR
 4336 016542  102402                       BVS     3$                      ;ERROR IF V SET
 4337 016544  001401                       BEQ     3$                      ;ERROR IF ZERO
 4338 016546  100401                       BMI     4$                      ;BRANCH IF GOOD
 4339 016550  104001             3$:       ERROR   +1              ;CPU ERROR
 4340                                                               ;SBC FAILED
 4341 016552                     4$:
 4342
 4343
 4346 016552                     MRLCC:
 4347
 4348                            ;       TEST ROL CONDITION CODES
 4349 016552  012704  060000               MOV     #60000,R4               ;R4= 0110000000000000
 4350 016556  000257                       CCC                             ;CC=0000
 4351 016560  006104                       ROL     R4                      ;R4= 1100000000000000
 4352 016562  103402                       BCS     1$                      ;ERROR IF CARRY
 4353 016564  102001                       BVC     1$                      ;ERROR IF V CLEAR
 4354 016566  100401                       BMI     2$                      ;BRANCH IF GOOD
 4355 016570  104001             1$:       ERROR   +1              ;CPU ERROR
 4356                                                               ;ROL  FAILED
 4357 016572  006104             2$:       ROL     R4                      ;R4= 1000000000000000
 4358 016574  103002                       BCC     3$                      ;ERROR IF CARRY CLEAR
 4359 016576  102401                       BVS     3$                      ;ERROR IF V SET
 4360 016600  100401                       BMI     4$                      ;BRANCH IF GOOD
```

```
********** DOUBLE OPERAND TESTS *****************

4361 016602  104001                  3$:     ERROR   +1              ;CPU ERROR
4362                                                                 ;BAD ROL
4363 016604  006104                  4$:     ROL     R4                      ;R4 = 0000000000000001
4364 016606  102003                          BVC     5$                      ;ERROR IF V CLEAR
4365 016610  103002                          BCC     5$                      ;ERROR IF C CLEAR
4366 016612  100401                          BMI     5$                      ;ERROR IF MINUS
4367 016614  001001                          BNE     6$                      ;BRANCH IF GOOD
4368 016616  104001                  5$:     ERROR   +1              ;CPU ERROR
4369                                                                 ;BAD ROL
4370 016620  006104                  6$:     ROL     R4                      ;R4=000000000000011
4371 016622  102402                          BVS     7$                      ;ERROR IF V SET
4372 016624  103401                          BCS     7$                      ;ERROR IF C SET
4373 016626  100001                          BPL     8$                      ;BRANCH IF GOOD
4374 016630  104001                  7$:     ERROR   +1              ;CPU ERROR
4375                                                                 ;BAD ROL
4376 016632                          8$:
4377
4378
4381 016632                          MRRCC:
4382
4383                                 ;       TEST ROR CONDITION CODES
4384 016632  012704  000003                  MOV     #3,R4                   ;R4= 0000000000000011
4385 016636  000257                          CCC                             ;CC= 0000
4386 016640  006004                          ROR     R4                      ;R4= 0000000000000001
4387 016642  103002                          BCC     1$                      ;ERROR IF NO CARRY
4388 016644  102001                          BVC     1$                      ;ERROR IF V CLEAR
4389 016646  100001                          BPL     2$                      ;BRANCH IF GOOD
4390 016650  104001                  1$:     ERROR   +1              ;CPU ERROR
4391                                                                 ;ROR  FAILED
4392 016652  006004                  2$:     ROR     R4                      ;R4= 1000000000000000
4393 016654  103002                          BCC     3$                      ;ERROR IF CARRY CLEAR
4394 016656  102401                          BVS     3$                      ;ERROR IF V SET
4395 016660  100401                          BMI     4$                      ;BRANCH IF GOOD
4396 016662  104001                  3$:     ERROR   +1              ;CPU ERROR
4397                                                                 ;BAD ROR
4398 016664  006004                  4$:     ROR     R4                      ;R4 = 1100000000000000
4399 016666  102002                          BVC     5$                      ;ERROR IF V
4400 016670  103401                          BCS     5$                      ;ERROR IF C SET
4401 016672  100401                          BMI     6$                      ;BRANCH IF GOOD
4402 016674  104001                  5$:     ERROR   +1              ;CPU ERROR
4403                                                                 ;BAD ROR
4404 016676  006004                  6$:     ROR     R4                      ;R4= 011000000000000
4405 016700  102402                          BVS     7$                      ;ERROR IF V SET
4406 016702  103401                          BCS     7$                      ;ERROR IF C SET
4407 016704  100001                          BPL     8$                      ;BRANCH IF GOOD
4408 016706  104001                  7$:     ERROR   +1              ;CPU ERROR
4409                                                                 ;BAD ROR
4410 016710                          8$:
4411
4412
4415 016710                          XCBIT:

4424
4425                                 ;       TEST C BIT WITH ROR/ROL
4426                                 ;;*****************************************************************
                                     ;THIS TEST IS TO CHECK FOR A SLOW C BIT PATH INTERNAL TO THE J11 DATA CHIP
                                     ;PROBLEM IS ONLY EXHIBITED ON EARLY MASK SETS (1590 AND 1593)
4427 016710  012701  052525                  MOV -   #52525,R1               ;INIT R1 WITH DATA
```

```
********** DOUBLE OPERAND TESTS *****************

4428 016714  000241                        CLC                         ;CLEAR THE C BIT
4429 016716  006001                        ROR     R1                  ;R1=025252, C BIT =1
4430 016720  006001                        ROR     R1                  ;R1=112525, C BIT =0
4431 016722  006001                        ROR     R1                  ;R1=045252, C BIT =1
4432 016724  103401                        BCS     1$                  ;BRANCH IF CARRY BIT SET
4433 016726  104001                        ERROR   +1              ;CPU ERROR
4434 016730  022701  045252        1$:     CMP     #45252,R1           ;IS DATA IN R1 = TO EXPECTED DATA?
4435 016734  001401                        BEQ     2$                  ;BRANCH IF YES
4436 016736  104001                        ERROR   +1              ;CPU ERROR
4437 016740  012701  125252        2$:     MOV     #125252,R1          ;SET UP R1
4438 016744  000241                        CLC                         ;CLEAR THE CARRY BIT
4439 016746  006101                        ROL     R1                  ;R1=052524, C BIT =1
4440 016750  006101                        ROL     R1                  ;R1=125251, C BIT =0
4441 016752  006101                        ROL     R1                  ;R1=052522, C BIT =1
4442 016754  103401                        BCS     3$                  ;BRANCH IF CARRY SET
4443 016756  104001                        ERROR   +1              ;CPU ERROR
4444 016760  022701  052522        3$:     CMP     #052522, R1         ;IS DATA IN R1 = TO EXPECTED DATA?
4445 016764  001401                        BEQ     4$
4446 016766  104001                        ERROR   +1              ;CPU ERROR
4447 016770                        4$:
4449
4451 016770                        MALCC:
4452
4453                               ;       TEST ASL CONDITION CODES
4454 016770  012704  060000                MOV     #60000,R4           ;R4= 0110000000000000
4455 016774  000257                        CCC                         ;CC=0000
4456 016776  006304                        ASL     R4                  ;C=0 R4= 1100000000000000
4457 017000  103402                        BCS     1$                  ;ERROR IF CARRY
4458 017002  102001                        BVC     1$                  ;ERROR IF V CLEAR
4459 017004  100401                        BMI     2$                  ;BRANCH IF GOOD
4460 017006  104001        1$:     ERROR   +1              ;CPU ERROR
4461                                                       ;ASL  FAILED
4462 017010  006304        2$:     ASL     R4                  ;C=1 R4= 1000000000000000
4463 017012  103002                BCC     3$                  ;ERROR IF CARRY CLEAR
4464 017014  102401                BVS     3$                  ;ERROR IF V SET
4465 017016  100401                BMI     4$                  ;BRANCH IF GOOD
4466 017020  104001        3$:     ERROR   +1              ;CPU ERROR
4467                                                       ;BAD ASL
4468 017022  006304        4$:     ASL     R4                  ;C=1 R4= 0000000000000000
4469 017024  102003                BVC     5$                  ;ERROR IF V CLEAR
4470 017026  103002                BCC     5$                  ;ERROR IF C CLEAR
4471 017030  100401                BMI     5$                  ;ERROR IF MINUS
4472 017032  001401                BEQ     6$                  ;BRANCH IF GOOD
4473 017034  104001        5$:     ERROR   +1              ;CPU ERROR
4474                                                       ;BAD ASL
4475 017036  006304        6$:     ASL     R4                  ;C=0 R4= 000000000000000
4476 017040  102402                BVS     7$                  ;ERROR IF V SET
4477 017042  103401                BCS     7$                  ;ERROR IF C SET
4478 017044  100001                BPL     8$                  ;BRANCH IF GOOD
4479 017046  104001        7$:     ERROR   +1              ;CPU ERROR
4480                                                       ;BAD ASL
4481 017050                        8$:
4482
4485 017050                        MARCC:
4486
4487                               ;       TEST ASR CONDITION CODES
4488 017050  012704  000341                MOV     #341,R4             ;R4= 0000000011100001
```

```
********** DOUBLE OPERAND TESTS *****************

 4489 017054  000257                      CCC                     ;CC=0000
 4490 017056  006204                      ASR     R4              ;R4= 0000000001110000
 4491 017060  103002                      BCC     1$              ;ERROR IF NO CARRY
 4492 017062  102001                      BVC     1$              ;ERROR IF V CLEAR
 4493 017064  100001                      BPL     2$              ;BRANCH IF GOOD
 4494 017066  104001            1$:       ERROR   +1          ;CPU ERROR
 4495                                                          ;ASR  FAILED
 4496 017070  052704  100001    2$:       BIS     #100001,R4      ;R4= 1000000001110001
 4497 017074  006204                      ASR     R4              ;R4= 1100000000111000
 4498 017076  103002                      BCC     3$              ;ERROR IF CARRY CLEAR
 4499 017100  102401                      BVS     3$              ;ERROR IF V SET
 4500 017102  100401                      BMI     4$              ;BRANCH IF GOOD
 4501 017104  104001            3$:       ERROR   +1          ;CPU ERROR
 4502                                                          ;BAD ASR
 4503 017106  006204            4$:       ASR     R4              ;R4= 1110000000011100
 4504 017110  102002                      BVC     5$              ;ERROR IF V
 4505 017112  103401                      BCS     5$              ;ERROR IF C SET
 4506 017114  100401                      BMI     6$              ;BRANCH IF GOOD
 4507 017116  104001            5$:       ERROR   +1          ;CPU ERROR
 4508                                                          ;BAD ASR
 4509 017120  006204            6$:       ASR     R4              ;R4= 1111000000001110
 4510 017122  102005                      BVC     7$              ;ERROR IF V CLEAR
 4511 017124  103404                      BCS     7$              ;ERROR IF C SET
 4512 017126  100003                      BPL     7$              ;ERROR IF PLUS
 4513 017130  022704  170016              CMP     #170016,R4      ;SEE IF EXPECTED RESULT
 4514 017134  001401                      BEQ     8$              ;BRANCH IF GOOD
 4515 017136  104001            7$:       ERROR   +1          ;CPU ERROR
 4516                                                          ;BAD ASR
 4517 017140                    8$:
 4518
 4519
 4522 017140                    MSXTCC:
 4523
 4524                           ;       TEST SXT CONDITION CODES / -*0-
 4525 017140  012704  123456              MOV     #123456,R4              ;R4=123456
 4526 017144  010401                      MOV     R4,R1               ;SAVE  CONTENTS
 4527 017146  000257                      CCC                         ;CC=0000
 4528 017150  006704                      SXT     R4                  ;R4=0 CC=0100
 4529 017152  103403                      BCS     1$                  ;ERROR IF CARRY
 4530 017154  100402                      BMI     1$                  ;ERROR IF MINUS
 4531 017156  102401                      BVS     1$                  ;ERROR IF OVERFLOW
 4532 017160  001401                      BEQ     2$                  ;BRANCH IF GOOD
 4533 017162  104001            1$:       ERROR   +1              ;CPU ERROR
 4534                                                              ;BAD SXT
 4535 017164  010104            2$:       MOV     R1,R4               ;RESTORE R4
 4536 017166  000277                      SCC                         ;CC=1111
 4537 017170  006704                      SXT     R4                  ;R4=-1 CC=1001
 4538 017172  001405                      BEQ     3$                  ;ERROR IF ZERO
 4539 017174  100004                      BPL     3$                  ;ERROR IF PLUS
 4540 017176  103003                      BCC     3$                  ;ERROR IF NO CARRY
 4541 017200  102402                      BVS     3$                  ;ERROR IF OVERFLOW
 4542 017202  005104                      COM     R4                  ;R4=0
 4543 017204  001401                      BEQ     4$                  ;BRANCH IF GOOD
 4544 017206  104001            3$:       ERROR   +1              ;CPU ERROR
 4545                                                              ;BAD SXT
 4546 017210                    4$:
 4547
```

```
********** DOUBLE OPERAND TESTS *****************

4548
4551 017210                            MXRCC:
4552
4553                                   ;       TEST XOR CONDITION CODES / **0-
4554 017210  012704  123456                      MOV     #123456,R4              ;R4=123456
4555 017214  012701  052525                      MOV     #52525,R1               ;R1=52525
4556 017220  000257                              CCC                             ;CC=0000
4557 017222  074104                              XOR     R1,R4                   ;*TI* R4=171173
4558 017224  102403                              BVS     1$                      ;;ERROR IF OVERFLOW
4559 017226  001402                              BEQ     1$                      ;ERROR IF ZERO
4560 017230  103401                              BCS     1$                      ;ERROR IF CARRY
4561 017232  100401                              BMI     2$                      ;BRANCH IF GOOD
4562 017234  104001                    1$:       ERROR   +1              ;CPU ERROR
4563                                                                     ;BAD XOR
4564 017236  012701  125252            2$:       MOV     #125252,R1              ;R1=125252
4565 017242  000277                              SCC                             ;CC=1111
4566 017244  074104                              XOR     R1,R4                   ;R4=054321
4567 017246  100403                              BMI     3$                      ;ERROR IF MINUS
4568 017250  001402                              BEQ     3$                      ;ERROR IF ZERO
4569 017252  103001                              BCC     3$                      ;ERROR IF CARRY CLEAR
4570 017254  102001                              BVC     4$                      ;BRANCH IF GOOD
4571 017256  104001                    3$:       ERROR   +1              ;CPU ERROR
4572                                                                     ;BAD XOR
4573 017260  074404                    4$:       XOR     R4,R4                   ;R4=0
4574 017262  102406                              BVS     5$                      ;ERROR IF OVREFLOW
4575 017264  100405                              BMI     5$                      ;ERROR IF MINUS
4576 017266  103004                              BCC     5$                      ;ERROR IF NO CARRY
4577 017270  001003                              BNE     5$                      ;ERROR IF NOT ZERO
4578 017272  022704  000000                      CMP     #0,R4                   ;SEE IF EXPECTED RESULT
4579 017276  001401                              BEQ     6$                      ;BRANCH IF GOOD
4580 017300  104001                    5$:       ERROR   +1              ;CPU ERROR
4581                                                                     ;BAD XOR
4582 017302                            6$:
4583
4584
4585                                   ;
4597 017302                            MSXT:
4598
4599                                   ;       TEST SXT (SIGN EXTEND INSTRUCTION)
4600                                   ;;*****************************************************************
                                       ;AN ADDITIONAL TEST IS INCLUDED TO CHECK FOR A SLOW N BIT PATH
                                       ;ON A TRANSITION FROM ZERO TO ONE INTERNAL TO THE J11 DATA CHIP
                                       ;THE PROBLEM IS ONLY EXHIBITED ON EARLY MASK SETS (1590 OR 1593)
4601 017302  005004                            CLR     R4                      ;TRASH R4
4602 017304  000257                            CCC                             ;CC=0000
4603 017306  000271                            .WORD   271                     ;CC=1001
4604 017310  006704                            SXT     R4                      ;*TEST INSTRUCTION
4605 017312  102405                            BVS     1$                      ;BRANCH IF OVERFLOW IS NOT CLEARED
4606 017314  100004                            BPL     1$                      ;BRANCH IF N BIT EFFECTED
4607 017316  001403                            BEQ     1$                      ;BRANNCH IF Z BIT EFFECTED
4608 017320  103002                            BCC     1$                      ;BRANCH IF C BIT EFFECTED
4609 017322  005204                            INC     R4
4610 017324  001401                            BEQ     2$                      ;BRANCH IF R4 =0
4611 017326  104001                    1$:     ERROR   +1                      ;CPU ERROR
                                                                ;CC SHOULD HAVE = 1101
4612
4613 017330  000277                    2$:     SCC
4614 017332  000250                            CLN                             ;CC=0111
```

```
********** DOUBLE OPERAND TESTS *****************

 4615 017334  005004                      CLR     R4
 4616 017336  012714  000055              MOV     #55,(R4)        ;TRASH R4
 4617 017342  006714                      SXT     (R4)            ;*TEST INSTRUCTION
 4618 017344  001005                      BNE     3$              ;BRANCH IF BIT EFFECTED
 4619 017346  102404                      BVS     3$              ;BRANCH IF OVERFLOW
 4620 017350  103403                      BCS     3$              ;
 4621 017352  100402                      BMI     3$              ;BRANCH IF N IS SET
 4622 017354  005714                      TST     (R4)            ;VERIFY INSTRUCTION WORKED
 4623 017356  001401                      BEQ     4$              ;BRANCH IF R4=0
 4624 017360  104001              3$:     ERROR   .+1             ;CPU ERROR
 4625                                                     ;SXT FAILED
 4626
 4627                              ;       NOW TEST FOR SLOW N BIT IN J11 DATA CHIP
 4628
 4629 017362  012700  177777      4$:     MOV     #-1,R0          ;R0=177777, N BIT = 1
 4630 017366  005004                      CLR     R4              ;CLEAT THE N BIT
 4631 017370  006700                      SXT     R0              ;****TEST INSTRUCTION***
 4632                                                             ;TEST N BIT TRANSITION 1 TO 0
 4633 017372  005700                      TST     R0              ;R0 SHOULD = 0
 4634 017374  001401                      BEQ     5$              ;BRANCH IF OK
 4635 017376  104001                      ERROR   .+1             ;CPU ERROR
 4636 017400  005000              5$:     CLR     R0              ;CLEAR R0, N BIT = 0
 4637 017402  012704  177777              MOV     #-1,R4          ;SET N BIT
 4638 017406  006700                      SXT     R0              ;***TEST INSTRUCTION***
 4639                                                             ;TEST N BIT TRANSITION 0 TO 1
 4640 017410  022700  177777              CMP     #-1,R0          ;R0 SHOULD = 177777
 4641 017414  001401                      BEQ     6$              ;BRANCH IF OK
 4642 017416  104001                      ERROR   .+1             ;CPU ERROR
 4643 017420                      6$:
 4645                             ;
 4647 017420                      MXOR:
 4648
 4649                             ;       TEST XOR
 4650 017420  012701  007643              MOV     #7643,R1                ;SETUP DATA
 4651 017424  012704  133333              MOV     #133333,R4              ;SETUP DATA
 4652 017430  000277                      SCC
 4653 017432  074401                      XOR     R4,R1                   ;*TEST INSTRUCTION
 4654 017434  100006                      BPL     1$                      ;BRANCH IF PLUS TO ERROR
 4655 017436  001405                      BEQ     1$                      ;ERROR IF ZERO
 4656 017440  103004                      BCC     1$                      ;ERROR IF CARRY CLEAR
 4657 017442  102403                      BVS     1$                      ;ERROR IF V SET
 4658 017444  020127  134570              CMP     R1,#134570              ;VERIFY CORRECT RESULT
 4659 017450  001401                      BEQ     2$                      ;BRANCH IF GOOD
 4660 017452  104001              1$:     ERROR   .+1             ;CPU ERROR
 4661                                                             ;BAD XOR
 4662 017454  010102              2$:     MOV     R1,R2
 4663 017456  000257                      CCC
 4664 017460  074402                      XOR     R4,R2                   ;*TEST INSTRUCTION
 4665 017462  100405                      BMI     3$                      ;ERROR IF MINUS
 4666 017464  102404                      BVS     3$                      ;ERROR IF OVERFLOW
 4667 017466  103403                      BCS     3$                      ;ERROR IF CARRY
 4668 017470  020227  007643              CMP     R2,#7643
 4669 017474  001401                      BEQ     4$                      ;BRANCH IF GOOD
 4670 017476  104001              3$:     ERROR   .+1             ;CPU ERROR
 4671                                                             ;BAD XOR
 4672 017500                      4$:
 4673
```

```
********** DOUBLE OPERAND TESTS *****************

                                        ;
017500                                  MSOB:
                                        ;       TEST SOB
017500  012704  000555                          MOV     #555,R4         ;SETUP TEST COUNTER
017504  000277                          SCC                             ;CC=17
017506  103015                  1$:     BCC     2$                      ;ERROR IF CARRY CLEAR
017510  102014                          BVC     2$                      ;ERROR IF NO OVERFLOW
017512  100013                          BPL     2$                      ;ERROR IF PLUS
017514  001012                          BNE     2$                      ;ERROR IF ZERO
017516  077405                          SOB     R4,1$                   ;*TEST INSTRUCTION
017520  103005                          BCC     3$                      ;ERROR IF CARRY CLEAR
017522  102004                          BVC     3$                      ;ERROR IF NO OVERFLOW
017524  100003                          BPL     3$                      ;ERROR IF PLUS
017526  001002                          BNE     3$                      ;ERROR IF ZERO
017530  000167  000010                  JMP     4$
017534  104001                  3$:     ERROR   +1                      ;CPU ERROR
                                                                ;CC EFFECTED DURING TEST
017536  000167  000002                  JMP     4$
017542  104001                  2$:     ERROR   +1                      ;CPU ERROR
                                                                ;CC EFFECTED AFTER TEST
017544  020427  000000          4$:     CMP     R4,#0                   ;IS R4 CORRECT
017550  001401                          BEQ     5$                      ;YES GO ON
017552  104001                          ERROR   +1                      ;CPU ERROR
                                                                ;NO GO TO ERROR
017554                          5$:

017554                          MMARK:

                                        ;       MARK INSTRUCTION TEST
017554  012706  000700                  MOV     #STBOT-100,SP           ;SETUP TEST STACK = 700
017560  012737  125252  000776          MOV     #125252,@#STBOT-2       ;SET UP NEW R5 VALUE ON STACK
017566  012705  017604                  MOV     #1$,R5                  ;PUT NEW PC IN R5
017572  012746  006437                  MOV     #MARK+37,-(SP)          ;INSERT MARK 37 INSTRUCTION ONTO STACK
017576  000277                          SCC
017600  000116                          JMP     (SP)                    ;* TEST INSTRUCTION
                                                                        ;MARK INSTRUCTION SHOULD HAVE GONE TO 1$
017602  104001                          ERROR   +1              ;CPU ERROR
                                        ;
017604  101002                  1$:     BHI     2$                      ;ERROR IF C OR Z BIT CLEAR
017606  100001                          BPL     2$                      ;ERROR IF N BIT CLEAR
017610  102401                          BVS     3$                      ;BRANCH IF V BIT SET
                                                                        ;BAD CONDITION CODES ON MARK
017612  104001                  2$:     ERROR   +1              ;CPU ERROR
017614  022705  125252          3$:     CMP     #125252,R5              ;VERIFY R5
017620  001401                          BEQ     4$                      ;BRANCH IF GOOD
017622  104001                          ERROR   +1              ;CPU ERROR

017624  020627  001000          4$:     CMP     SP,#STBOT               ;VERIFY THAT STACK IS CORRECT
017630  001401                          BEQ     15$                     ;BRANCH IF OK
                                                                        ;ERROR! STACK WAS NOT CORRECT AFTER MARK
017632  104001                          ERROR   +1              ;CPU ERROR

017634  012746  052525          15$:    MOV     #52525,-(SP)            ;SETUP EXPECTED R5
017640  012746  006400                  MOV     #6400,-(SP)             ;MOVE MARK 0 INSTRUCTION ON STACK
017644  010605                          MOV     SP,R5                   ;R5=ADDRESS OF INSTRUCTION
```

```
********** DOUBLE OPERAND TESTS *****************

 4735 017646  004767  000004                JSR     PC,5$                   ;LEAVE 6$ ON STACK
 4736 017652  000167  000006        6$:     JMP     16$                     ;MARK RETURNED CORRECTLY
 4737
 4738 017656  000257                5$:     CCC                             ;CLEAR THE CONDITION CODES
 4739 017660  000205                        RTS     R5                      ;RETURN TO MARK INSTRUCTION
 4740                                                                        ;NEXT INSTRUCTION ON STACK IS THE RETURN
 4741                                                                        ;FROM THE JSR
 4742                                                                        ;ERROR! BAD MARK SEQUENCE
 4743 017662  104001                        ERROR   ^1              ;CPU ERROR
 4744
 4745 017664  101402                16$:    BLOS    7$                      ;IS C OR Z BIT SET?
 4746 017666  100401                        BMI     7$                      ;IS N BIT SET?
 4747 017670  102001                        BVC     8$                      ;IS V BIT SET?
 4748                                                                        ;ERROR! CONDITIONS CODES INCORRECT
 4749 017672  104001                7$:     ERROR   ^1              ;CPU ERROR
 4750
 4751 017674  020627  001000        8$:     CMP     R6,#STBOT               ;IS THE STACK CORRECT?
 4752 017700  001401                        BEQ     9$                      ;BRANCH IF YES
 4753                                                                        ;ERROR! BAD STACK CLEANUP
 4754 017702  104001                        ERROR   ^1              ;CPU ERROR
 4755 017704  022705  052525        9$:     CMP     #52525,R5               ;VERIFY THAT R5 WAS LOADED PROPERLY.
 4756 017710  001401                        BEQ     10$                     ;IF OK, GO TO NEXT TEST.
 4757                                                                        ;ERROR! R5 WAS NOT CORRECT AFTER MARK.
 4758 017712  104001                        ERROR   ^1              ;CPU ERROR
 4759 017714                        10$:
 4761 017714                        XME100:
 4763
 4764                               ;       TEST CLEAR CONDITION CODES INSTRUCTION
 4765 017714  012737  030017  177776        MOV     #30017,@#177776         ;SETUP PSW
 4766 017722  000257                        CCC                             ; TEST INSTRUCTION
 4767 017724  022737  030000  177776        CMP     #30000,@#177776         ;DID IT CLEAR ALL CONDITION CODE BITS
 4768 017732  001401                        BEQ     1$                      ;YES GO ON
 4769 017734  104001                        ERROR   ^1              ;CPU ERROR
 4770                                                                        ;NO GO TO ERROR
 4771 017736                        1$:
 4772                               ;
 4774 017736                        XME101:
 4776
 4777                               ;       TEST CLEAR C BIT INSTRUCTION
 4778 017736  012737  030017  177776        MOV     #30017,@#177776         ;SETUP PSW
 4779 017744  000241                        CLC                             ; TEST INSTRUCTION
 4780 017746  022737  030016  177776        CMP     #30016,@#177776         ;DID IT CLEAR CARRY BIT
 4781 017754  001401                        BEQ     1$                      ;YES GO ON
 4782                                                                        ;C BIT NOT CLEAR GO TO ERROR
 4783 017756  104001                        ERROR   ^1              ;CPU ERROR
 4784 017760                        1$:
 4785                               ;
 4788 017760                        TE102:
 4789                               ;       TEST CLEAR N BIT INST (CLN)
 4790 017760  012737  030017  177776        MOV     #30017,@#177776         ;SETUP PSW
 4791 017766  000250                        CLN                             ; TEST INSTRUCTION
 4792 017770  022737  030007  177776        CMP     #30007,@#177776         ;DID IT CLEAR NEGATIVE BIT
 4793 017776  001401                        BEQ     1$                      ;YES GO ON
 4794 020000  104001                        ERROR   ^1              ;CPU ERROR
 4795                                                                        ;NO GO TO ERROR
 4796 020002                        1$:
 4797                               ;
```

```
********** DOUBLE OPERAND TESTS ******************

4800 020002                          TE103:
4801                                 ;       TEST CLEAR V BIT INST (CLV)
4802 020002  012737  030017  177776          MOV     #30017,@#177776         ;SETUP PSW
4803 020010  000242                          CLV                             ; TEST INSTRUCTION
4804 020012  022737  030015  177776          CMP     #30015,@#177776         ;DID IT CLEAR OVERFLOW BIT
4805 020020  001401                          BEQ     1$                      ;YES GO ON
4806 020022  104001                          ERROR   +1              ;CPU ERROR
4807                                                                 ;NO GO TO ERROR
4808 020024                          1$:
4809                                 ;
4812 020024                          TE104:
4813                                 ;       TEST CLEAR Z BIT INST (CLZ)
4814 020024  012737  030017  177776          MOV     #30017,@#177776         ;SETUP PSW
4815 020032  000244                          CLZ                             ; TEST INSTRUCTION
4816 020034  022737  030013  177776          CMP     #30013,@#177776         ;DID IT CLEAR ZERO BIT
4817 020042  001401                          BEQ     1$                      ;YES GO ON
4818 020044  104001                          ERROR   +1              ;CPU ERROR
4819                                                                 ;NO GO TO ERROR
4820 020046                          1$:
4821                                 ;
4824 020046                          TE105:
4825                                 ;       TEST SET CONDITION CODES INST (SCC)
4826 020046  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
4827 020054  000277                          SCC                             ; TEST INSTRUCTION
4828 020056  022737  030017  177776          CMP     #30017,@#177776         ;DID IT SET ALL CONDITION CODE BITS
4829 020064  001401                          BEQ     1$                      ;YES GO ON
4830 020066  104001                          ERROR   +1              ;CPU ERROR
4831                                                                 ;NO GO TO ERROR
4832 020070                          1$:
4833                                 ;
4836 020070                          TE106:
4837                                 ;       TEST SET C BIT INST (SEC)
4838 020070  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
4839 020076  000261                          SEC                             ; TEST INSTRUCTION
4840 020100  022737  030001  177776          CMP     #30001,@#177776         ;DID IT SET THE CARRY BIT
4841 020106  001401                          BEQ     1$                      ;YES GO ON
4842 020110  104001                          ERROR   +1              ;CPU ERROR
4843                                                                 ;NO GO TO ERROR
4844 020112                          1$:
4845                                 ;
4848 020112                          TE107:
4849                                 ;       TEST SET N BIT INST (SEN)
4850 020112  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
4851 020120  000270                          SEN                             ; TEST INSTRUCTION
4852 020122  022737  030010  177776          CMP     #30010,@#177776         ;DID IT SET THE NEGATIVE BIT
4853 020130  001401                          BEQ     1$                      ;YES GO ON
4854 020132  104001                          ERROR   +1              ;CPU ERROR
4855                                                                 ;NO GO TO ERROR
4856 020134                          1$:
4857                                 ;
4860 020134                          TE110:
4861                                 ;       TEST SET V BIT INST (SEV)
4862 020134  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
4863 020142  000262                          SEV                             ; TEST INSTRUCTION
4864 020144  022737  030002  177776          CMP     #30002,@#177776         ;DID IT SET THE OVERFLOW BIT
4865 020152  001401                          BEQ     1$                      ;YES GO ON
4866 020154  104001                          ERROR   +1              ;CPU ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

                                                                ;NO GO TO ERROR
                                1$:
                                ;
                                TE111:
                                ;       TEST SET Z BIT INST (SEZ)
020156  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
020164  000264                          SEZ                             ; TEST INSTRUCTION
020166  022737  030004  177776          CMP     #30004,@#177776         ;DID IT SET THE ZERO BIT
020174  001401                          BEQ     1$                      ;YES GO ON
020176  104001                          ERROR   .+1                 ;CPU ERROR
                                                                ;NO GO TO ERROR
020200                          1$:
                                ;
                                TE112:
                                ;       TEST MULTIPLE CLEARS OF CC BITS
020200  012737  030000  177776          MOV     #30000,@#177776         ;INIT PSW
020206  000277                          SCC                             ;SETUP PSW
020210  000243                          .WORD   243                     ; TEST CLC CLV
020212  022737  030014  177776          CMP     #30014,@#177776         ;PSW CORRECT?
020220  001401                          BEQ     1$                      ;YES GO ON
020222  104001                          ERROR   .+1                 ;CPU ERROR
                                                                ;NO GO TO ERROR
020224  000277                  1$:     SCC                             ;SETUP PSW
020226  000245                          .WORD   245                     ; TEST CLC CLZ
020230  022737  030012  177776          CMP     #30012,@#177776         ;PSW CORRECT?
020236  001401                          BEQ     2$                      ;YES GO ON
020240  104001                          ERROR   .+1                 ;CPU ERROR
                                                                ;NO GO TO ERROR
020242  000277                  2$:     SCC                             ;SETUP PSW
020244  000246                          .WORD   246                     ; TEST CLV CLZ
020246  022737  030011  177776          CMP     #30011,@#177776         ;PSW CORRECT?
020254  001401                          BEQ     3$                      ;YES GO ON
020256  104001                          ERROR   .+1                 ;CPU ERROR
                                                                ;NO GO TO ERROR
020260  000277                  3$:     SCC                             ;SETUP PSW
020262  000247                          .WORD   247                     ; TEST CLC CLV CLZ
020264  022737  030010  177776          CMP     #30010,@#177776         ;PSW CORRECT?
020272  001401                          BEQ     4$                      ;YES GO ON
020274  104001                          ERROR   .+1                 ;CPU ERROR
                                                                ;NO GO TO ERROR
020276  000277                  4$:     SCC                             ;SETUP PSW
020300  000251                          .WORD   251                     ; TEST CLN CLC
020302  022737  030006  177776          CMP     #30006,@#177776         ;PSW CORRECT?
020310  001401                          BEQ     5$                      ;YES GO ON
020312  104001                          ERROR   .+1                 ;CPU ERROR
                                                                ;NO GO TO ERROR
020314  000277                  5$:     SCC                             ;SETUP PSW
020316  000252                          .WORD   252                     ; TEST CLN CLV
020320  022737  030005  177776          CMP     #30005,@#177776         ;PSW CORRECT?
020326  001401                          BEQ     6$                      ;YES GO ON
020330  104001                          ERROR   .+1                 ;CPU ERROR
                                                                ;NO GO TO ERROR
020332  000277                  6$:     SCC                             ;SETUP PSW
020334  000253                          .WORD   253                     ; TEST CLN CLC CLV
020336  022737  030004  177776          CMP     #30004,@#177776         ;PSW CORRECT?
020344  001401                          BEQ     7$                      ;YES GO ON
020346  104001                          ERROR   .+1                 ;CPU ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

 4928
 4929 020350  000277                    7$:     SCC                             ;NO GO TO ERROR
 4930 020352  000254                            .WORD   254                             ; TEST CLN CLZ
 4931 020354  022737  030003  177776            CMP     #30003,@#177776         ;PSW CORRECT?
 4932 020362  001401                            BEQ     8$                      ;YES GO ON
 4933 020364  104001                            ERROR   .1              ;CPU ERROR
 4934                                                                   ;NO GO TO ERROR
 4935 020366  000277                    8$:     SCC                             ;SETUP PSW
 4936 020370  000255                            .WORD   255                             ; TEST CLN CLC CLZ
 4937 020372  022737  030002  177776            CMP     #30002,@#177776         ;PSW CORRECT?
 4938 020400  001401                            BEQ     9$                      ;YES GO ON
 4939 020402  104001                            ERROR   .1              ;CPU ERROR
 4940                                                                   ;NO GO TO ERROR
 4941 020404  000277                    9$:     SCC                             ;SETUP PSW
 4942 020406  000256                            .WORD   256                             ; TEST CLN CLV CLZ
 4943 020410  022737  030001  177776            CMP     #30001,@#177776         ;SETUP PSW
 4944 020416  001401                            BEQ     10$                     ;YES GO ON
 4945 020420  104001                            ERROR   .1              ;CPU ERROR
 4946                                                                   ;NO GO TO ERROR
 4947 020422                            10$:
 4948                                   ;
 4951 020422                            TE113:
 4952                                   ;       TEST MULTIPLE SETS OF CC BITS
 4953 020422  012737  030000  177776            MOV     #30000,@#177776         ;INIT PSW
 4954 020430  000263                            .WORD   263                             ; TEST SEC SEV
 4955 020432  022737  030003  177776            CMP     #30003,@#177776         ;PSW CORRECT?
 4956 020440  001401                            BEQ     1$                      ;YES GO ON
 4957 020442  104001                            ERROR   .1              ;CPU ERROR
 4958                                                                   ;NO GO TO ERROR
 4959 020444  000257                    1$:     CCC                             ;SETUP PSW
 4960 020446  000265                            .WORD   265                             ; TEST SEC SEZ
 4961 020450  022737  030005  177776            CMP     #30005,@#177776         ;PSW CORRECT?
 4962 020456  001401                            BEQ     2$                      ;YES GO ON
 4963 020460  104001                            ERROR   .1              ;CPU ERROR
 4964                                                                   ;NO GO TO ERROR
 4965 020462  000257                    2$:     CCC                             ;SETUP PSW
 4966 020464  000266                            .WORD   266                             ; TEST SEV SEZ
 4967 020466  022737  030006  177776            CMP     #30006,@#177776         ;PSW CORRECT?
 4968 020474  001401                            BEQ     3$                      ;YES GO ON
 4969 020476  104001                            ERROR   .1              ;CPU ERROR
 4970                                                                   ;NO GO TO ERROR
 4971 020500  000257                    3$:     CCC                             ;SETUP PSW
 4972 020502  000267                            .WORD   267                             ; TEST SEC SEV SEZ
 4973 020504  022737  030007  177776            CMP     #30007,@#177776         ;PSW CORRECT?
 4974 020512  001401                            BEQ     4$                      ;YES GO ON
 4975 020514  104001                            ERROR   .1              ;CPU ERROR
 4976                                                                   ;NO GO TO ERROR
 4977 020516  000257                    4$:     CCC                             ;SETUP PSW
 4978 020520  000271                            .WORD   271                             ; TEST SEN SEC
 4979 020522  022737  030011  177776            CMP     #30011,@#177776         ;PSW CORRECT?
 4980 020530  001401                            BEQ     5$                      ;YES GO ON
 4981 020532  104001                            ERROR   .1              ;CPU ERROR
 4982                                                                   ;NO GO TO ERROR
 4983 020534  000257                    5$:     CCC                             ;SETUP PSW
 4984 020536  000272                            .WORD   272                             ; TEST SEN SEV
 4985 020540  022737  030012  177776            CMP     #30012,@#177776         ;PSW CORRECT?
 4986 020546  001401                            BEQ     6$                      ;YES GO ON
```

```
********** DOUBLE OPERAND TESTS *****************

 4987 020550  104001                      ERROR   +1              ;CPU ERROR
 4988                                                              ;NO GO TO ERROR
 4989 020552  000257              6$:      CCC                             ;SETUP PSW
 4990 020554  000273                       .WORD   273                     ; TEST SEN SEC SEV
 4991 020556  022737  030013  177776       CMP     #30013,@#177776         ;PSW CORRECT?
 4992 020564  001401                       BEQ     7$                      ;YES GO ON
 4993 020566  104001                       ERROR   +1              ;CPU ERROR
 4994                                                              ;NO GO TO ERROR
 4995 020570  000257              7$:      CCC                             ;SETUP PSW
 4996 020572  000274                       .WORD   274                     ; TEST SEN SEZ
 4997 020574  022737  030014  177776       CMP     #30014,@#177776         ;PSW CORRECT?
 4998 020602  001401                       BEQ     8$                      ;YES GO ON
 4999 020604  104001                       ERROR   +1              ;CPU ERROR
 5000                                                              ;NO GO TO ERROR
 5001 020606  000257              8$:      CCC                             ;SETUP PSW
 5002 020610  000275                       .WORD   275                     ; TEST SEN SEC SEZ
 5003 020612  022737  030015  177776       CMP     #30015,@#177776         ;PSW CORRECT?
 5004 020620  001401                       BEQ     9$                      ;YES GO ON
 5005 020622  104001                       ERROR   +1              ;CPU ERROR
 5006                                                              ;NO GO TO ERROR
 5007 020624  000257              9$:      CCC                             ;SETUP PSW
 5008 020626  000276                       .WORD   276                     ; TEST SEN SEV SEZ
 5009 020630  022737  030016  177776       CMP     #30016,@#177776         ;PSW CORRECT?
 5010 020636  001401                       BEQ     10$                     ;YES GO ON
 5011 020640  104001                       ERROR   +1              ;CPU ERROR
 5012                                                              ;NO GO TO ERROR
 5013 020642                      10$:
 5014                             ;
 5017 020642                      TE113A:
 5018                             ;        TEST SIGNED AND CONDITIONAL BRANCHES
 5019 020642  000257                       CCC                             ;CLEAR ALL CC BITS IN PSW
 5020 020644  002001                       BGE     1$                      ;BGE SHOULD BRANCH
 5021 020646  104001                       ERROR   +1              ;CPU ERROR
 5022                                                              ;ERROR; DIDN'T BRANCH
 5023 020650  003001              1$:      BGT     2$                      ;BGT SHOULD BRANCH
 5024 020652  104001                       ERROR   +1              ;CPU ERROR
 5025                                                              ;ERROR;  DIDN'T BRANCH
 5026 020654  003401              2$:      BLE     3$                      ;BLE SHOULDN'T BRANCH
 5027 020656  000401                       BR      4$                      ;BRANCH TO NEXT TEST
 5028 020660  104001              3$:      ERROR   +1              ;CPU ERROR
 5029                                                              ;ERROR; BLE SHOULD NOT HAVE BRANCHED
 5030 020662  002401              4$:      BLT     5$                      ;BLT SHOULD NOT BRANCH
 5031 020664  000401                       BR      6$                      ;BRANCH TO NEXT TEST
 5032 020666  104001              5$:      ERROR   +1              ;CPU ERROR
 5033                                                              ;ERROR; BLT SHOULD NOT HAVE BRANCHED
 5034 020670  000264              6$:      SEZ                             ;SET THE Z BIT IN PSW
 5035 020672  003401                       BLE     7$                      ;BLE SHOULD BRANCH
 5036 020674  104001                       ERROR   +1              ;CPU ERROR
 5037                                                              ;ERROR; BLE DIDN'T BRANCH
 5038 020676  003001              7$:      BGT     8$                      ;BGT SHOULD NOT BRANCH
 5039 020700  000401                       BR      9$                      ;BRANCH TO NEXT TEST
 5040 020702  104001              8$:      ERROR   +1              ;CPU ERROR
 5041                                                              ;ERROR; BGT SHOULD NOT HAVE BRANCHED
 5042 020704  000257              9$:      CCC                             ;CLEAR ALL CC BITS IN PSW
 5043 020706  000270                       SEN                             ;SET N BIT IN PSW
 5044 020710  002401                       BLT     10$                     ;SHOULD BRANCH TO NEXT TEST
 5045 020712  104001                       ERROR   +1              ;CPU ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

 5046
 5047 020714  003401                10$:  BLE     11$          ;ERROR; BLT SHOULD HAVE BRANCHED
 5048 020716  104001                      ERROR   +1                   ;SHOULD BRANCH TO NEXT TEST
 5049                                                          ;CPU ERROR
 5050 020720  002001                11$:  BGE     12$          ;ERROR; BLE SHOULD HAVE BRANCHED
 5051 020722  000401                      BR      13$                  ;BGE SHOULD NOT BRANCH
 5052 020724  104001                12$:  ERROR   +1                   ;BRANCH TO NEXT TEST
 5053                                                          ;CPU ERROR
 5054 020726  003001                13$:  BGT     14$          ;ERROR; BGE SHOULD NOT HAVE BRANCHED
 5055 020730  000401                      BR      15$                  ;BGT SHOULD NOT BRANCH
 5056 020732  104001                14$:  ERROR   +1                   ;BRANCH TO NEXT TEST
 5057                                                          ;CPU ERROR
 5058 020734  000257                15$:  CCC                  ;ERROR; BGT SHOULD NOT HAVE BRANCHED
 5059 020736  000262                      SEV                          ;CLEAR ALL CC BITS
 5060 020740  003401                      BLE     16$                  ;SET V BIT IN PSW
 5061 020742  104001                      ERROR   +1                   ;BLE SHOULD BRANCH
 5062                                                          ;CPU ERROR
 5063 020744  002401                16$:  BLT     17$          ;ERROR; BLE DIDN'T BRANCH
 5064 020746  104001                      ERROR   +1                   ;BLT SHOULD BRANCH
 5065                                                          ;CPU ERROR
 5066 020750  002001                17$:  BGE     18$          ;ERROR; BLT DIDN'T BRANCH
 5067 020752  000401                      BR      19$                  ;BGE SHOULDN'T BRANCH
 5068 020754  104001                18$:  ERROR   +1                   ;BRANCH TO NEXT TEST
 5069                                                          ;CPU ERROR
 5070 020756  003001                19$:  BGT     20$          ;ERROR; BGE SHOULD NOT HAVE BRANCHED
 5071 020760  000401                      BR      21$                  ;BGT SHOULD NOT BRANCH
 5072 020762  104001                20$:  ERROR   +1                   ;BRANCH TO NEXT TEST
 5073                                                          ;CPU ERROR
 5074 020764  000257                21$:  CCC                  ;ERROR; BGT SHOULD NOT HAVE BRANCHED
 5075 020766  000272                      .WORD   272                  ;CLEAR ALL CC BITS
 5076 020770  002001                      BGE     22$                  ;SET N AND V BITS IN PSW
 5077 020772  104001                      ERROR   +1                   ;BGE SHOULD BRANCH
 5078                                                          ;CPU ERROR
 5079 020774  003001                22$:  BGT     23$          ;ERROR; BGE DIDN'T BRANCH
 5080 020776  104001                      ERROR   +1                   ;BGT SHOULD BRANCH
 5081                                                          ;CPU ERROR
 5082 021000  003401                23$:  BLE     24$          ;ERROR; BGT DIDN'T BRANCH
 5083 021002  000401                      BR      25$                  ;BLE SHOULDN'T BRANCH
 5084 021004  104001                24$:  ERROR   +1                   ;BRANCH TO NEXT TEST
 5085                                                          ;CPU ERROR
 5086 021006  002401                25$:  BLT     26$          ;ERROR; BLE SHOULD NOT HAVE BRANCHED
 5087 021010  000401                      BR      27$                  ;BLT SHOULD NOT BRANCH
 5088 021012  104001                26$:  ERROR   +1                   ;BRANCH TO NEXT TEST
 5089                                                          ;CPU ERROR
 5090 021014                        27$:                       ;ERROR; BLT SHOULD NOT HAVE BRANCHED
 5093 021014                        TE114:
 5094                               ;     TEST NOP INST
 5095 021014  012737  030000 177776       MOV     #30000,@#177776      ;INIT PSW
 5096 021022  012700  000001              MOV     #1,R0                ;INIT R0
 5097 021026  012701  000002              MOV     #2,R1                ;INIT R1
 5098 021032  012702  000003              MOV     #3,R2                ;INIT R2
 5099 021036  012703  000004              MOV     #4,R3                ;INIT R3
 5100 021042  012704  000005              MOV     #5,R4                ;INIT R4
 5101 021046  012705  000006              MOV     #6,R5                ;INIT R5
 5102 021052  010637  003012              MOV     R6,@#SLOC00          ;SAVE SP
 5103 021056  012737  030017 114146       MOV     #30017,@#EXPDAT      ;SETUP PSW
 5104 021064  000277                      SCC                          ;SET ALL CONDITION CODE BITS
```

```
********** DOUBLE OPERAND TESTS *****************

 5105 021066  000240                          NOP                             ; TEST INSTRUCTION
 5106 021070  000240                          NOP                             ; TEST INSTRUCTION
 5107 021072  000240                          NOP                             ; TEST INSTRUCTION
 5108 021074  004767  000046                  JSR     PC,T114                 ;CHECK PSW, AND GPR'S
 5109 021100  020637  003012                  CMP     R6,@#SLOC00             ;CHECK SP
 5110 021104  001401                          BEQ     1$                      ;OK GO ON
 5111 021106  104001                          ERROR   +1                  ;CPU ERROR
 5112                                                                     ;NO GO TO ERROR
 5113 021110  012737  030000  114146  1$:     MOV     #30000,@#EXPDAT         ;SETUP PSW
 5114 021116  000257                          CCC                             ;CLEAR ALL CONDITION CODE BITS
 5115 021120  000260                          .WORD   260                     ; TEST FOR NOP OPERATION
 5116 021122  000260                          .WORD   260                     ; TEST FOR NOP OPERATION
 5117 021124  000260                          .WORD   260                     ; TEST FOR NOP OPERATION
 5118 021126  004767  000014                  JSR     PC,T114                 ;CHECK PSW, AND GPR'S
 5119 021132  020637  003012                  CMP     R6,@#SLOC00             ;CHECK SP
 5120 021136  001401                          BEQ     2$                      ;OK GO ON
 5121 021140  104001                          ERROR   +1                  ;CPU ERROR
 5122                                                                     ;NO GO TO ERROR
 5123 021142                          2$:
 5124 021142  000167  000104                  JMP     FINNOP
 5125                                 ;
 5126 021146  023737  114146  177776  T114:   CMP     @#EXPDAT,@#177776       ;CHECK PSW
 5127 021154  001405                          BEQ     TA114                   ;OK GO ON
 5128 021156  010067  157216                  MOV     R0,400                  ;SAVE R0
 5129 021162  104001                          ERROR   +1                  ;CPU ERROR
 5130                                                                     ;NO GO TO ERROR
 5131 021164  016700  157210                  MOV     400,R0                  ;RESTORE R0
 5132 021170  022700  000001          TA114:  CMP     #1,R0                   ;CHECK R0
 5133 021174  001401                          BEQ     TB114                   ;OK GO ON
 5134 021176  104001                          ERROR   +1                  ;CPU ERROR
 5135                                                                     ;NO GO TO ERROR
 5136 021200  022701  000002          TB114:  CMP     #2,R1                   ;CHECK R1
 5137 021204  001401                          BEQ     TC114                   ;OK GO ON
 5138 021206  104001                          ERROR   +1                  ;CPU ERROR
 5139                                                                     ;NO GO TO ERROR
 5140 021210  022702  000003          TC114:  CMP     #3,R2                   ;CHECK R2
 5141 021214  001401                          BEQ     TD114                   ;OK GO ON
 5142 021216  104001                          ERROR   +1                  ;CPU ERROR
 5143                                                                     ;NO GO TO ERROR
 5144 021220  022703  000004          TD114:  CMP     #4,R3                   ;CHECK R3
 5145 021224  001401                          BEQ     TF114                   ;OK GO ON
 5146 021226  104001                          ERROR   +1                  ;CPU ERROR
 5147                                                                     ;NO GO TO ERROR
 5148 021230  022704  000005          TF114:  CMP     #5,R4                   ;CHECK R4
 5149 021234  001401                          BEQ     TG114                   ;OK GO ON
 5150 021236  104001                          ERROR   +1                  ;CPU ERROR
 5151                                                                     ;NO GO TO ERROR
 5152 021240  022705  000006          TG114:  CMP     #6,R5                   ;CHECK R5
 5153 021244  001401                          BEQ     TH114                   ;OK GO ON
 5154 021246  104001                          ERROR   +1                  ;CPU ERROR
 5155                                                                     ;NO GO TO ERROR
 5156 021250  000207                  TH114:  RTS     PC                      ;RETURN
 5157                                 ;
 5158 021252  000240                  FINNOP: NOP
 5159                                 ;
 5162                                 ;
 5163                                 ;--------------------------------
```

```
********** DOUBLE OPERAND TESTS *****************

                                ;TEST ATERNATE REGISTER SET
                                ;
021254  005000                        CLR     R0              ;------CLEAR------
021256  005001                        CLR     R1              ;-----PRIMARY----
021260  005002                        CLR     R2              ;-----GENERAL-----
021262  005003                        CLR     R3              ;-----PURPOSE-----
021264  005004                        CLR     R4              ;-----REGISTER----
021266  005005                        CLR     R5              ;-------SET-------
021270  012767  004000  156500        MOV     #4000,PS        ;SELECT ALTERNATE REGISTER SET
021276  032767  004000  156472        BIT     #BIT11,PS       ;TEST TO SEE THAT BIT IS SET IN PS
021304  001001                        BNE     1$              ;IF IT'S SET GO TESTS REGISTERS
021306  104001                        ERROR   +1      ;CPU ERROR
                                ;ERROR! ALTERNATE REGISTER SELECT
                                ;BIT IN PS IS NOT SET
021310  012700  177777        1$:     MOV     #177777,R0      ;WRITE ALL ONES TO ALTERNATE REG SET
021314  010001                        MOV     R0,R1           ;
021316  010102                        MOV     R1,R2           ;
021320  010203                        MOV     R2,R3           ;
021322  010304                        MOV     R3,R4           ;
021324  010405                        MOV     R4,R5           ;
021326  042767  004000  156442        BIC     #BIT11,PS       ;CLEAR BIT 11 IN PS
021334  032767  004000  156434        BIT     #BIT11,PS       ;IS BIT 11 = 0?
021342  001401                        BEQ     2$              ;IF IT'S NOT ZERO
021344  104001                        ERROR   +1      ;CPU ERROR
                                ;ERROR! BIT 11 DID NOT CLEAR
021346  005700                2$:     TST     R0              ;MAKE SURE THAT WE REALLY
021350  001401                        BEQ     3$              ;WERE TALKING TO ALTERNATE REGISTERS.
021352  104001                        ERROR   +1      ;CPU ERROR
                                ;ERROR!
021354  005701                3$:     TST     R1              ;
021356  001401                        BEQ     4$
021360  104001                        ERROR   +1      ;CPU ERROR
                                ;ERROR!
021362  005702                4$:     TST     R2              ;
021364  001401                        BEQ     5$              ;
021366  104001                        ERROR   +1      ;CPU ERROR
                                ;ERROR!
021370  005703                5$:     TST     R3              ;
021372  001401                        BEQ     6$              ;
021374  104001                        ERROR   +1      ;CPU ERROR
                                ;
021376  005704                6$:     TST     R4              ;
021400  001401                        BEQ     7$              ;
021402  104001                        ERROR   +1      ;CPU ERROR
                                ;
021404  005705                7$:     TST     R5              ;
021406  001401                        BEQ     8$              ;
021410  104001                        ERROR   +1      ;CPU ERROR
                                ;ERROR!
021412  012767  004000  156356  8$:     MOV     #BIT11,PS       ;ENABLE ALTERNATE REGISTER SET
021420  005000                        CLR     R0              ;-----CLEAR-----
021422  005001                        CLR     R1              ;---ALTERNATE---
021424  005002                        CLR     R2              ;----REGISTER---
021426  005003                        CLR     R3              ;------SET-----
021430  005004                        CLR     R4              ;
021432  005005                        CLR     R5              ;
021434  042767  004000  156334        BIC     #BIT11,PS       ;BACK TO PRIMARY GPRS
```

```
********** DOUBLE OPERAND TESTS *****************

 5221 021442  012700  177777          MOV     #177777,R0          ;ENSURE THAT WRITES TO PRIMARY GPRS
 5222 021446  010001                  MOV     R0,R1               ;DO NOT EFFECT ALTERNATE GPRS
 5223 021450  010102                  MOV     R1,R2               ;
 5224 021452  010203                  MOV     R2,R3               ;
 5225 021454  010304                  MOV     R3,R4               ;
 5226 021456  010405                  MOV     R4,R5               ;
 5227 021460  052767  004000  156310  BIS     #BIT11,PS           ;RETURN TO ALTERNATE REGISTERS
 5228 021466  005700                  TST     R0                  ;SHOULD BE ALL 0'S
 5229 021470  001401                  BEQ     9$                  ;
 5230 021472  104001                  ERROR   +1              ;CPU ERROR
 5231                                                         ;ERROR!
 5232 021474  005701          9$:     TST     R1                  ;
 5233 021476  001401                  BEQ     10$                 ;
 5234 021500  104001                  ERROR   +1              ;CPU ERROR
 5235                                                         ;
 5236 021502  005702          10$:    TST     R2                  ;
 5237 021504  001401                  BEQ     11$                 ;
 5238 021506  104001                  ERROR   +1              ;CPU ERROR
 5239                                                         ;
 5240 021510  005703          11$:    TST     R3                  ;
 5241 021512  001401                  BEQ     12$                 ;
 5242 021514  104001                  ERROR   +1              ;CPU ERROR
 5243                                                         ;
 5244 021516  005704          12$:    TST     R4                  ;
 5245 021520  001401                  BEQ     13$                 ;
 5246 021522  104001                  ERROR   +1              ;CPU ERROR
 5247                                                         ;
 5248 021524  005705          13$:    TST     R5                  ;
 5249 021526  001401                  BEQ     14$                 ;
 5250 021530  104001                  ERROR   +1              ;CPU ERROR
 5251 021532                  14$:
 5252
 5253 021532                  ALROTS:
 5254                         ;       ALTERNATE REGISTER SET R0 BIT TESTS
 5255 021532  012700  177777          MOV     #177777,R0              ;R0=177777
 5256 021536  020027  177777          CMP     R0,#177777              ;DOES R0=177777
 5257 021542  001401                  BEQ     1$                      ;YES GO ON
 5258 021544  104001                  ERROR   +1              ;CPU ERROR
 5259                                                                 ;NO GO TO ERROR
 5260 021546  005000          1$:     CLR     R0                      ;R0=0
 5261 021550  020027  000000          CMP     R0,#0                   ;DOES R0=0
 5262 021554  001401                  BEQ     2$                      ;YES GO ON
 5263 021556  104001                  ERROR   +1              ;CPU ERROR
 5264                                                                 ;NO GO TO ERROR
 5265 021560  012700  125252  2$:     MOV     #125252,R0              ;R0=125252
 5266 021564  020027  125252          CMP     R0,#125252              ;DOES R0=125252
 5267 021570  001401                  BEQ     3$                      ;YES GO ON
 5268 021572  104001                  ERROR   +1              ;CPU ERROR
 5269                                                                 ;NO GO TO ERROR
 5270 021574  012700  052525  3$:     MOV     #52525,R0               ;R0=52525
 5271 021600  020027  052525          CMP     R0,#52525               ;DOES R0=52525
 5272 021604  001401                  BEQ     4$                      ;YES GO ON
 5273 021606  104001                  ERROR   +1              ;CPU ERROR
 5274                                                                 ;NO GO TO ERROR
 5275 021610                  4$:
 5276                         ;
 5279 021610                  ALR1TS:
```

********** DOUBLE OPERAND TESTS **********

```
5280                                    ;       ALTERNATE REGISTER SET R1 BIT TESTS
5281 021610  012701  177777                    MOV     #177777,R1              ;R1=177777
5282 021614  020127  177777                    CMP     R1,#177777              ;DOES R1=177777
5283 021620  001401                            BEQ     1$                      ;YES GO ON
5284 021622  104001                            ERROR   +1              ;CPU ERROR
5285                                                                           ;NO GO TO ERROR
5286 021624  005001                    1$:     CLR     R1                      ;R1=0
5287 021626  020127  000000                    CMP     R1,#0                   ;DOES R1=0
5288 021632  001401                            BEQ     2$                      ;YES GO ON
5289 021634  104001                            ERROR   +1              ;CPU ERROR
5290                                                                           ;NO GO TO ERROR
5291 021636  012701  125252            2$:     MOV     #125252,R1              ;R1=125252
5292 021642  020127  125252                    CMP     R1,#125252              ;DOES R1=125252
5293 021646  001401                            BEQ     3$                      ;YES GO ON
5294 021650  104001                            ERROR   +1              ;CPU ERROR
5295                                                                           ;NO GO TO ERROR
5296 021652  012701  052525            3$:     MOV     #52525,R1               ;R1=52525
5297 021656  020127  052525                    CMP     R1,#52525               ;DOES R1=52525
5298 021662  001401                            BEQ     4$                      ;YES GO ON
5299 021664  104001                            ERROR   +1              ;CPU ERROR
5300                                                                           ;NO GO TO ERROR
5301 021666                            4$:
5302                                    ;
5305 021666                            ALR2TS:
5306                                    ;       ALTERNATE REGISTER SET R2 BIT TESTS
5307 021666  012702  177777                    MOV     #177777,R2              ;R2=177777
5308 021672  020227  177777                    CMP     R2,#177777              ;DOES R2=177777
5309 021676  001401                            BEQ     1$                      ;YES GO ON
5310 021700  104001                            ERROR   +1              ;CPU ERROR
5311                                                                           ;NO GO TO ERROR
5312 021702  005002                    1$:     CLR     R2                      ;R2=0
5313 021704  020227  000000                    CMP     R2,#0                   ;DOES R2=0
5314 021710  001401                            BEQ     2$                      ;YES GO ON
5315 021712  104001                            ERROR   +1              ;CPU ERROR
5316                                                                           ;NO GO TO ERROR
5317 021714  012702  125252            2$:     MOV     #125252,R2              ;R2=125252
5318 021720  020227  125252                    CMP     R2,#125252              ;DOES R2=125252
5319 021724  001401                            BEQ     3$                      ;YES GO ON
5320 021726  104001                            ERROR   +1              ;CPU ERROR
5321                                                                           ;NO GO TO ERROR
5322 021730  012702  052525            3$:     MOV     #52525,R2               ;R2=52525
5323 021734  020227  052525                    CMP     R2,#52525               ;DOES R2=52525
5324 021740  001401                            BEQ     4$                      ;YES GO ON
5325 021742  104001                            ERROR   +1              ;CPU ERROR
5326                                                                           ;NO GO TO ERROR
5327 021744                            4$:
5328                                    ;
5331 021744                            ALR3TS:
5332                                    ;       ALTERNATE REGISTER SET R3 BIT TESTS
5333 021744  012703  177777                    MOV     #177777,R3              ;R3=177777
5334 021750  020327  177777                    CMP     R3,#177777              ;DOES R3=177777
5335 021754  001401                            BEQ     1$                      ;YES GO ON
5336 021756  104001                            ERROR   +1              ;CPU ERROR
5337                                                                           ;NO GO TO ERROR
5338 021760  005003                    1$:     CLR     R3                      ;R3=0
5339 021762  020327  000000                    CMP     R3,#0                   ;DOES R3=0
5340 021766  001401                            BEQ     2$                      ;YES GO ON
```

```
********** DOUBLE OPERAND TESTS *****************

 5341 021770  104001                        ERROR   +1              ;CPU ERROR
 5342                                                                 ;NO GO TO ERROR
 5343 021772  012703  125252        2$:     MOV     #125252,R3      ;R3=125252
 5344 021776  020327  125252                CMP     R3,#125252      ;DOES R3=125252
 5345 022002  001401                        BEQ     3$              ;YES GO ON
 5346 022004  104001                        ERROR   +1              ;CPU ERROR
 5347                                                                 ;NO GO TO ERROR
 5348 022006  012703  052525        3$:     MOV     #52525,R3       ;R3=52525
 5349 022012  020327  052525                CMP     R3,#52525       ;DOES R3=52525
 5350 022016  001401                        BEQ     4$              ;YES GO ON
 5351 022020  104001                        ERROR   +1              ;CPU ERROR
 5352                                                                 ;NO GO TO ERROR
 5353 022022                        4$:
 5354                                ;
 5357 022022                        ALR4TS:
 5358                                ;       ALTERNATE REGISTER SET R4 BIT TESTS
 5359 022022  012704  177777                MOV     #177777,R4      ;R4=177777
 5360 022026  020427  177777                CMP     R4,#177777      ;DOES R4=177777
 5361 022032  001401                        BEQ     1$              ;YES GO ON
 5362 022034  104001                        ERROR   +1              ;CPU ERROR
 5363                                                                 ;NO GO TO ERROR
 5364 022036  005004                1$:     CLR     R4              ;R4=0
 5365 022040  020427  000000                CMP     R4,#0           ;DOES R4=0
 5366 022044  001401                        BEQ     2$              ;YES GO ON
 5367 022046  104001                        ERROR   +1              ;CPU ERROR
 5368                                                                 ;NO GO TO ERROR
 5369 022050  012704  125252        2$:     MOV     #125252,R4      ;R4=125252
 5370 022054  020427  125252                CMP     R4,#125252      ;DOES R4=125252
 5371 022060  001401                        BEQ     3$              ;YES GO ON
 5372 022062  104001                        ERROR   +1              ;CPU ERROR
 5373                                                                 ;NO GO TO ERROR
 5374 022064  012704  052525        3$:     MOV     #52525,R4       ;R4=52525
 5375 022070  020427  052525                CMP     R4,#52525       ;DOES R4=52525
 5376 022074  001401                        BEQ     4$              ;YES GO ON
 5377 022076  104001                        ERROR   +1              ;CPU ERROR
 5378                                                                 ;NO GO TO ERROR
 5379 022100                        4$:
 5380                                ;
 5383 022100                        ALR5TS:
 5384                                ;       ALTERNATE REGISTER SET R5 BIT TESTS
 5385 022100  012705  177777                MOV     #177777,R5      ;R5=177777
 5386 022104  020527  177777                CMP     R5,#177777      ;DOES R5=177777
 5387 022110  001401                        BEQ     1$              ;YES GO ON
 5388 022112  104001                        ERROR   +1              ;CPU ERROR
 5389                                                                 ;NO GO TO ERROR
 5390 022114  005005                1$:     CLR     R5              ;R5=0
 5391 022116  020527  000000                CMP     R5,#0           ;DOES R5=0
 5392 022122  001401                        BEQ     2$              ;YES GO ON
 5393 022124  104001                        ERROR   +1              ;CPU ERROR
 5394                                                                 ;NO GO TO ERROR
 5395 022126  012705  125252        2$:     MOV     #125252,R5      ;R5=125252
 5396 022132  020527  125252                CMP     R5,#125252      ;DOES R5=125252
 5397 022136  001401                        BEQ     3$              ;YES GO ON
 5398 022140  104001                        ERROR   +1              ;CPU ERROR
 5399                                                                 ;NO GO TO ERROR
 5400 022142  012705  052525        3$:     MOV     #52525,R5       ;R5=52525
 5401 022146  020527  052525                CMP     R5,#52525       ;DOES R5=52525
```

```
********** DOUBLE OPERAND TESTS *****************

 5402 022152  001401                        BEQ     4$                      ;YES GO ON
 5403 022154  104001                        ERROR   ;1              ;CPU ERROR
 5404                                                                ;NO GO TO ERROR
 5405 022156  042767  004000  155612  4$:   BIC     #BIT11,PS               ;RETURN TO PRIMARY GEN PURPOSE REGS
 5406
 5407                               ;
 5410 022164                        TE115:
 5411                               ;       TEST MOVE FROM PROCCESSOR STATUS INST (MFPS)
 5412 022164  005004                        CLR     R4                      ;SETUP DESTINATION R4
 5413
 5414 022166  012701  022374                MOV     #TE115A,R1              ;SETUP POINTERS TO TABLES
 5415 022172  012702  022404                MOV     #TE115B,R2              ;
 5416 022176  012703  022412                MOV     #TE115C,R3              ;
 5417 022202  012137  177776        1$:     MOV     (R1)+,@#177776          ;SETUP PSW
 5418 022206  106704                        MFPS    R4                      ; TEST INSTRUCTION
 5419 022210  023722  177776                CMP     @#177776,(R2)+          ;CHECK PSW
 5420 022214  001401                        BEQ     2$                      ;OK GO ON
 5421 022216  104001                        ERROR   ;1              ;CPU ERROR
 5422                                                                ;NO GO TO ERROR
 5423 022220  020423                2$:     CMP     R4,(R3)+                ;CHECK R4
 5424 022222  001401                        BEQ     3$                      ;OK GO ON
 5425 022224  104001                        ERROR   ;1              ;CPU ERROR
 5426                                                                ;NO GO TO ERROR
 5427 022226  021127  177777        3$:     CMP     (R1),#177777            ;ARE WE DONE
 5428 022232  001363                        BNE     1$                      ;NO GO TO 1$
 5429
 5430
 5431 022234  012701  022374                MOV     #TE115A,R1              ;SETUP POINTERS TO TABLES
 5432 022240  012702  022404                MOV     #TE115B,R2              ;
 5433 022244  012703  022412                MOV     #TE115C,R3              ;
 5434 022250  010605                        MOV     R6,R5                   ;SAVE STACK IN R5
 5435 022252  011137  177776        101$:   MOV     (R1),@#177776           ;SETUP PSW
 5436 022256  106706                        MFPS    R6                      ; TEST INSTRUCTION
 5437 022260  023712  177776                CMP     @#177776,(R2)           ;CHECK PSW
 5438 022264  001401                        BEQ     102$                    ;OK GO ON
 5439 022266  104001                        ERROR   ;1              ;CPU ERROR
 5440                                                                ;NO GO TO ERROR
 5441 022270  020613                102$:   CMP     R6,(R3)                 ;CHECK R6
 5442 022272  001401                        BEQ     103$                    ;OK GO ON
 5443 022274  104001                        ERROR   ;1              ;CPU ERROR
 5444                                                                ;NO GO TO ERROR
 5445 022276  010506                103$:   MOV     R5,R6                   ;RESTORE STACK
 5446
 5447
 5448 022300  012701  022374                MOV     #TE115A,R1              ;SETUP POINTERS TO TABLES
 5449 022304  012702  022404                MOV     #TE115B,R2              ;
 5450 022310  012703  022420                MOV     #TE115D,R3              ;
 5451 022314  005037  114146                CLR     @#EXPDAT                ;INIT EXPECTED DATA HOLDER.
 5452 022320  012704  114146        4$:     MOV     #EXPDAT,R4              ;SETUP POINTER TO TEST LOCATION
 5453 022324  012137  177776                MOV     (R1)+,@#177776          ;SETUP PSW
 5454 022330  106724                        MFPS    (R4)+                   ; TEST INSTRUCTION
 5455 022332  023722  177776                CMP     @#177776,(R2)+          ;CHECK PSW
 5456 022336  001401                        BEQ     5$                      ;OK GO ON
 5457 022340  104001                        ERROR   ;1              ;CPU ERROR
 5458                                                                ;NO GO TO ERROR
 5459 022342  020427  114147        5$:     CMP     R4,#EXPDAT+1            ;CHECK R4
 5460 022346  001401                        BEQ     6$                      ;OK GO ON
```

```
********** DOUBLE OPERAND TESTS *****************

5461 022350  104001                          ERROR   +1              ;CPU ERROR
5462                                                                 ;NO GO TO ERROR
5463 022352  023723  114146          6$:     CMP     @#EXPDAT,(R3)+          ;CHECK TEST LOCATION
5464 022356  001401                          BEQ     7$              ;OK GO ON
5465 022360  104001                          ERROR   +1              ;CPU ERROR
5466                                                                 ;NO GO TO ERROR
5467 022362  021127  177777          7$:     CMP     (R1),#177777            ;ARE WE DONE
5468 022366  001354                          BNE     4$              ;NO GO TO 4$
5469
5470 022370  000167  000032                  JMP     TE115F
5471
5472                                 ;
5473                                 ;
5474 022374  030207                  TE115A: .WORD   30207
5475 022376  030000                          .WORD   30000
5476 022400  030057                          .WORD   30057
5477 022402  177777                          .WORD   177777
5478 022404  030211                  TE115B: .WORD   30211
5479 022406  030004                          .WORD   30004
5480 022410  030041                          .WORD   30041
5481 022412  177607                  TE115C: .WORD   177607
5482 022414  000000                          .WORD   0
5483 022416  000057                          .WORD   57
5484 022420  000207                  TE115D: .WORD   207
5485 022422  000000                          .WORD   0
5486 022424  000057                          .WORD   57
5487 022426  000240                  TE115F: NOP
5488                                 ;
5491 022430                          TE116:
5492                                 ;       TEST MOVE TO PROCESSOR STATUS INST (MTPS)
5493
5494 022430  012737  030000  177776          MOV     #30000,@#177776         ;SET PSW TO KERNEL MODE
5495 022436  012701  022732                  MOV     #TE116D,R1              ;SETUP POINTERS TO TABLES
5496 022442  012702  022710                  MOV     #TE116B,R2              ;
5497 022446  010103                          MOV     R1,R3                   ;
5498 022450  004767  000136                  JSR     PC,T116                 ; TEST INSTRUCTION AND CHECK PSW
5499                                                                         ;AND SOURCE OPERAND
5500
5501
5502 022454  012737  140000  177776          MOV     #140000,@#177776        ;SET PSW TO USER MODE
5503 022462  012706  000600                  MOV     #600,R6                 ;SETUP USER STACK
5504 022466  012701  022732                  MOV     #TE116D,R1              ;SETUP POINTERS TO TABLES
5505 022472  012702  022722                  MOV     #TE116C,R2              ;
5506 022476  010103                          MOV     R1,R3                   ;
5507 022500  004767  000106                  JSR     PC,T116                 ; TEST INSTRUCTION AND CHECK PSW
5508                                                                         ;AND SOURCE OPERAND
5509
5510
5511 022504  012737  140000  177776          MOV     #140000,@#177776        ;SET PSW TO USER MODE AND CLEAR CC BITS
5512 022512  012701  022704                  MOV     #TE116A,R1              ;SETUP POINTERS TO TABLES
5513 022516  012702  022722                  MOV     #TE116C,R2              ;
5514 022522  012703  022732                  MOV     #TE116D,R3              ;
5515 022526  010104                          MOV     R1,R4                   ;SAVE A COPY OF R1 INTO R4
5516 022530  004767  000110                  JSR     PC,TA116                ; TEST INSTRUCTION AND CHECK PSW
5517                                                                         ;AND SOURCE OPERAND
5518
5519
```

```
********** DOUBLE OPERAND TESTS *****************

5520 022534  012737  030000  177776          MOV     #30000,@#177776         ;SET PSW TO KERNEL MODE
5521 022542  012701  022704                  MOV     #TE116A,R1              ;SETUP POINTERS TO TABLES
5522 022546  012702  022710                  MOV     #TE116B,R2              ;
5523 022552  012703  022732                  MOV     #TE116D,R3              ;
5524 022556  010104                          MOV     R1,R4                   ;SAVE A COPY OF R1 INTO R4
5525 022560  004767  000060                  JSR     PC,TA116                ; TEST INSTRUCTION AND CHECK PSW
5526                                                                         ;AND SOURCE OPERAND
5527
5528 022564  005037  177776                  CLR     @#177776                ;SET PSW TO KERNEL MODE
5529 022570  106427  177412                  MTPS    #177412                 ;TEST INSTRUCTION
5530 022574  022737  000012  177776          CMP     #12,@#177776            ;IS PSW CORRECT
5531 022602  001401                          BEQ     100$                    ;YES GO ON
5532 022604  104001                          ERROR   +1              ;CPU ERROR
5533                                                                 ;NO GO TO ERROR
5534 022606                          100$:
5535 022606  000167  000130                  JMP     FIN116
5536
5537 022612  012105                  T116:   MOV     (R1)+,R5                ;MOVE TEST DATA TO R5
5538 022614  106405                          MTPS    R5                      ; TEST INSTRUCTION
5539 022616  023722  177776                  CMP     @#177776,(R2)+          ;IS PSW CORRECT
5540 022622  001401                          BEQ     1$                      ;YES GO ON
5541 022624  104001                          ERROR   +1              ;CPU ERROR
5542                                                                 ;NO GO TO ERROR
5543 022626  022305                  1$:     CMP     (R3)+,R5                ;IS R5 CORRECT
5544 022630  001401                          BEQ     2$                      ;YES GO ON
5545 022632  104001                          ERROR   +1              ;CPU ERROR
5546                                                                 ;NO GO TO ERROR
5547 022634  021227  177777          2$:     CMP     (R2),#177777            ;ARE WE DONE
5548 022640  001364                          BNE     T116                    ;NO GO TO T116
5549 022642  000207                          RTS     PC                      ;RETURN
5550
5551 022644  106421                  TA116:  MTPS    (R1)+                   ; TEST INSTRUCTION
5552 022646  023722  177776                  CMP     @#177776,(R2)+          ;IS PSW CORRECT
5553 022652  001401                          BEQ     1$                      ;YES GO ON
5554 022654  104001                          ERROR   +1              ;CPU ERROR
5555                                                                 ;NO GO TO ERROR
5556 022656  122423                  1$:     CMPB    (R4)+,(R3)+             ;CHECK TEST LOCATION
5557 022660  001401                          BEQ     2$                      ;OK GO ON
5558 022662  104001                          ERROR   +1              ;CPU ERROR
5559                                                                 ;NO GO TO ERROR
5560 022664  020104                  2$:     CMP     R1,R4                   ;IS SOURCE OPERAND CORRECT
5561 022666  001401                          BEQ     3$                      ;YES GO ON
5562 022670  104001                          ERROR   +1              ;CPU ERROR
5563                                                                 ;NO GO TO ERROR
5564 022672  005203                  3$:     INC     R3                      ;POINT TO NEXT WORD
5565 022674  021227  177777                  CMP     (R2),#177777            ;ARE WE DONE
5566 022700  001361                          BNE     TA116                   ;NO GO TO TA116
5567 022702  000207                          RTS     PC                      ;RETURN
5568
5569 022704     377                  TE116A: .BYTE   377
5570 022705     000                          .BYTE   0
5571 022706     252                          .BYTE   252
5572 022707     125                          .BYTE   125
5573 022710  030357                  TE116B: .WORD   30357
5574 022712  030000                          .WORD   30000
5575 022714  030252                          .WORD   30252
5576 022716  030105                          .WORD   30105
```

```
********** DOUBLE OPERAND TESTS ******************

 5577 022720  177777                        .WORD   177777
 5578 022722  140017              TE116C:   .WORD   140017
 5579 022724  140000                        .WORD   140000
 5580 022726  140012                        .WORD   140012
 5581 022730  140005                        .WORD   140005
 5582 022732  177777              TE116D:   .WORD   177777
 5583 022734  177400                        .WORD   177400
 5584 022736  177652                        .WORD   177652
 5585 022740  177525                        .WORD   177525
 5586                             ;
 5587 022742                      FIN116:
 5588                             ;
 5598 022742                      TE117:
 5599                             ;       TEST MFPT (MOVE FROM PROCESSOR TYPE)
 5600 022742  013746  000010              MOV     @#10,-(SP)              ;SAVE VECTOR
 5601 022746  012737  023054  000010      MOV     #TE117A,@#10            ;SETUP VECTOR TO HANDLE POSSIBLE ILLEGAL INST TRAP
 5602 022754  012700  177777              MOV     #177777,R0              ;INIT R0
 5603 022760  012737  030000  177776      MOV     #30000,@#177776         ;SETUP PSW
 5604 022766  000007                      .WORD   7                       ; TEST INSTRUCTION
 5605 022770  022737  030000  177776      CMP     #30000,@#177776         ;IS PSW CORRECT
 5606 022776  001401                      BEQ     1$                      ;YES GO ON
 5607 023000  104001                      ERROR   .+1                 ;CPU ERROR
 5608                                                                 ;NO GO TO ERROR
 5609 023002  020027  000005      1$:     CMP     R0,#5                   ;IS R0 CORRECT
 5610 023006  001401                      BEQ     2$                      ;YES GO ON
 5611 023010  104001                      ERROR   .+1                 ;CPU ERROR
 5612                                                                 ;NO GO TO ERROR
 5613 023012  012700  177777      2$:     MOV     #177777,R0              ;INIT R0
 5614 023016  000277                      SCC                             ;SET ALL CC BITS
 5615 023020  000007                      .WORD   7                       ; TEST INSTRUCTION
 5616 023022  022737  030017  177776      CMP     #30017,@#177776         ;IS PSW CORRECT
 5617 023030  001401                      BEQ     3$                      ;YES GO ON
 5618 023032  104001                      ERROR   .+1                 ;CPU ERROR
 5619                                                                 ;NO GO TO ERROR
 5620 023034  020027  000005      3$:     CMP     R0,#5                   ;IS R0 CORRECT
 5621 023040  001401                      BEQ     4$                      ;YES GO ON
 5622 023042  104001                      ERROR   .+1                 ;CPU ERROR
 5623                                                                 ;NO GO TO ERROR
 5624 023044  012637  000010      4$:     MOV     (SP)+,@#10              ;RESTORE VECTOR
 5625
 5626 023050  000167  000002              JMP     FIN117
 5627                             ;
 5628 023054  104001              TE117A: ERROR   .+1                 ;CPU ERROR
 5629                                                                 ;GO TO ERROR IF TRAP TAKES PLACE
 5630                             ;
 5631 023056  000240              FIN117: NOP
 5632                             ;
 5644 023060                      TE120:
 5645                             ;       TEST HALT (NOT KERNEL MODE)
 5646 023060  005037  177766              CLR     @#177766                ;INIT CPU ERROR REG
 5647 023064  005037  177776              CLR     @#177776                ;INIT PSW-SET KERNEL MODE
 5648 023070  013746  000004              MOV     @#4,-(SP)               ;SAVE VECTOR
 5649 023074  013746  000006              MOV     @#6,-(SP)               ;SAVE VECTOR
 5650 023100  012737  023134  000004      MOV     #TE120A,@#4             ;SET UP VECTOR TO HANDLE ILLEGAL HALT
 5651 023106  005037  000006              CLR     @#6                     ;SET UP VECTOR TO COME BACK IN KERNEL MODE
 5652 023112  012767  140000  154656      MOV     #140000,PS              ;SET IN USER MODE
 5653 023120  012706  000600              MOV     #600,R6                 ;INITIALIZE THE USER STACK POINTER
```

```
********** DOUBLE OPERAND TESTS *****************

 5654 023124  000000                        HALT                            ; TEST INSTRUCTION
 5655 023126  104001                PROCNT: ERROR   +1                      ;CPU ERROR
 5656                                                                       ;IF NOTHING HAPPENED GO TO ERROR
 5657 023130  000167  000044                JMP     FIN120                  ;
 5658                               ;
 5659                               ;
 5660 023134  022737  030000 177776 TE120A: CMP     #30000,@#177776         ;IS PSW CORRECT/PREVIOUS MODE = USER?
 5661 023142  001401                        BEQ     1$                      ;YES GO ON
 5662 023144  104001                        ERROR   +1                      ;CPU ERROR
 5663                                                                       ;NO GO TO ERROR
 5664 023146  022737  000200 177766 1$:     CMP     #200,@#177766           ; TEST CPU ERROR REGISTER
 5665 023154  001401                        BEQ     2$                      ;YES GO ON
 5666 023156  104001                        ERROR   +1                      ;CPU ERROR
 5667                                                                       ;NO GO TO ERROR
 5668 023160  022627  023126          2$:   CMP     (SP)+,#PROCNT           ;DOES STACK CONTAIN CORRECT PC
 5669 023164  001401                        BEQ     3$                      ;YES GO ON
 5670 023166  104001                        ERROR   +1                      ;CPU ERROR
 5671                                                                       ;NO GO TO ERROR
 5672 023170  022627  140000          3$:   CMP     (SP)+,#140000           ;DOES STACK CONTAIN CORRECT PSW
 5673 023174  001401                        BEQ     FIN120                  ;YES GO ON
 5674 023176  104001                        ERROR   +1                      ;CPU ERROR
 5675                                                                       ;NO GO TO ERROR
 5676 023200  012637  000006        FIN120: MOV     (SP)+,@#6               ;RESTORE VECTOR
 5677 023204  012637  000004                MOV     (SP)+,@#4               ;RESTORE VECTOR
 5678
 5679                               ;
 5682 023210                        TE121:
 5683                               ;       TEST RESET
 5684                               ;       CMPB    #APTENV,$ENV            ;ARE WE IN APT MODE?
 5685                               ;       BNE     1$                      ;IF NOT: DO THIS TEST
 5686 023210  005767  155772                TST     $PASS                   ;FIRST PASS??
 5687 023214  001402                        BEQ     1$                      ;IF YES, DO IT
 5688 023216  000167  000622                JMP     FIN122                  ;ELSE SKIP THIS TEST BECAUSE RESETS
 5689                                                                       ;SCREW UP THE APT MONITOR.
 5690 023222  012737  030340 177776 1$:     MOV     #30340,@#177776         ;SETUP PSW TO KERNEL MODE
 5691 023230  012737  160000 177572         MOV     #160000,@#177572        ;SETUP MMR0
 5692 023236  012737  000077 172516         MOV     #77,@#172516            ;SETUP MMR3
 5693 023244  005037  177772                CLR     @#177772                ;CLEAR PIRQ
 5694 023250  023727  177772 000000         CMP     @#177772,#0             ;IS PIRQ CORRECT
 5695 023256  001401                        BEQ     C121A                   ;YES GO ON
 5696 023260  104001                        ERROR   +1                      ;CPU ERROR
 5697                                                                       ;NO GO TO ERROR
 5698 023262  012737  025000 177772 C121A: MOV     #25000,@#177772         ;MOVE AN ALTERNATING PATTERN TO PIRQ
 5699 023270  022737  025252 177772         CMP     #25252,@#177772         ;IS PIRQ CORRECT
 5700 023276  001401                        BEQ     C121B                   ;YES GO ON
 5701 023300  104001                        ERROR   +1                      ;CPU ERROR
 5702                                                                       ;NO GO TO ERROR
 5703 023302  012737  077000 177772 C121B: MOV     #77000,@#177772         ;SETUP PIRQ
 5704 023310  022737  077314 177772         CMP     #77314,@#177772         ;IS PIRQ CORRECT
 5705 023316  001401                        BEQ     C121C                   ;YES GO ON
 5706 023320  104001                        ERROR   +1                      ;CPU ERROR
 5707                                                                       ;NO GO TO ERROR
 5708 023322  000277                C121C:  SCC                             ;SET ALL CC BITS
 5709 023324  000005                        RESET                           ; TEST INSTRUCTION
 5710 023326  022737  030357 177776         CMP     #30357,@#177776         ;IS PSW CORRECT
 5711 023334  001401                        BEQ     1$                      ;YES GO ON
 5712 023336  104001                        ERROR   +1                      ;CPU ERROR
```

********** DOUBLE OPERAND TESTS **********

```
5713
5714 023340  013701  177572          1$:   MOV     @#SR0,R1          ;NO GO TO ERROR
5715 023344  042701  000176                BIC     #176,R1                 ;SAVE SR0 IN R1.
5716 023350  022701  000000                CMP     #0,R1                   ;STRIP OFF UNDEFINED BITS 1-6 FROM MMR0
5717 023354  001401                        BEQ     2$                      ;IS MMR0 CORRECT
5718 023356  104001                        ERROR   .+1                     ;YES GO ON
5719                                                                ;CPU ERROR
5720 023360  022737  000000  172516  2$:   CMP     #0,@#172516       ;NO GO TO ERROR
5721 023366  001401                        BEQ     3$                      ;IS MMR3 CORRECT
5722 023370  104001                        ERROR   .+1                     ;YES GO ON
5723                                                                ;CPU ERROR
5724 023372  022737  000000  177772  3$:   CMP     #0,@#177772       ;NO GO TO ERROR
5725 023400  001401                        BEQ     4$                      ;IS PIRQ CORRECT
5726 023402  104001                        ERROR   .+1                     ;YES GO ON
5727                                                                ;CPU ERROR
5728                                                                ;NO GO TO ERROR
5729 023404  013702  000004          4$:   MOV     @#4,R2                  ;SAVE LOC 4 IN R2
5730 023410  013703  000006                MOV     @#6,R3                  ;SAVE LOC 6 IN R3
5731 023414  012737  023552  000004        MOV     #8$,@#4                 ;SETUP VECTORS IN CASE AN ERROR CAUSES
5732                                                                ;A HALT TO OCCUR IN USER MODE.
5733 023422  012737  000340  000006        MOV     #340,@#6                ;THIS SETS KERNEL MODE, AND PREVENTS PRIQ
5734                                                                ;INTERRUPTS FROM TRASHING THE STACK IF A
5735                                                                ;USER MODE HALT OCCURS.
5736 023430  012737  140340  177776        MOV     #140340,@#177776        ;SETUP PSW TO USER MODE
5737 023436  012737  160000  177572        MOV     #160000,@#177572        ;SETUP MMR0
5738 023444  012737  000077  172516        MOV     #77,@#172516            ;SETUP MMR3
5739 023452  012737  077000  177772        MOV     #77000,@#177772         ;SETUP PIRQ
5740 023460  000277                        SCC                             ;SET ALL CC BITS
5741 023462  000005                        RESET                           ; TEST INSTRUCTION
5742 023464  022737  140357  177776        CMP     #140357,@#177776        ;IS PSW CORRECT
5743 023472  001402                        BEQ     5$                      ;YES GO ON
5744 023474  000000                        HALT                            ;USER MODE HALT; WILL TRAP TO LOC 4
5745 023476  104001                        ERROR   .+1             ;CPU ERROR
5746                                                                ;NO GO TO ERROR
5747 023500  013701  177572          5$:   MOV     @#177572,R1             ;SAVE MMR0 IN R1
5748 023504  042701  000176                BIC     #176,R1                 ;CLEAR BITS 1-6 FROM MMR0
5749 023510  022701  160000                CMP     #160000,R1              ;IS MMR0 CORRECT
5750 023514  001402                        BEQ     6$                      ;YES GO ON
5751 023516  000000                        HALT                            ;USER MODE HALT; WILL TRAP TO LOC 4
5752 023520  104001                        ERROR   .+1             ;CPU ERROR
5753                                                                ;NO GO TO ERROR
5754 023522  022737  000077  172516  6$:   CMP     #77,@#172516            ;IS MMR3 CORRECT
5755 023530  001402                        BEQ     7$                      ;YES GO ON
5756 023532  000000                        HALT                            ;USER MODE HALT; WILL TRAP TO LOC 4
5757 023534  104001                        ERROR   .+1             ;CPU ERROR
5758                                                                ;NO GO TO ERROR
5759 023536  022737  077314  177772  7$:   CMP     #77314,@#177772         ;IS PIRQ CORRECT
5760 023544  001403                        BEQ     9$                      ;YES GO ON
5761 023546  000000                        HALT                            ;USER MODE HALT; WILL TRAP TO LOC 4
5762 023550  104001                        ERROR   .+1             ;CPU ERROR
5763                                                                ;NO GO TO ERROR
5764 023552  000002                  8$:   RTI                             ;USER MODE HALT OCCURRED; GO TO ERROR.
5765
5766 023554  005037  177772          9$:   CLR     @#177772                ;CLEAR PIRQ
5767 023560  005037  177776                CLR     @#177776                ;CLEAR PSW
5768 023564  010237  000004                MOV     R2,@#4                  ;RESTORE VECTORS TO PREVIOUS STATE
5769 023570  010337  000006                MOV     R3,@#6                  ;
```

```
********** DOUBLE OPERAND TESTS *****************

 5770 023574                          FIN121:
 5771                                 ;
 5774 023574                          TE122:
 5775                                 ;       TEST SPL (SET PRIORITY LEVEL)
 5776
 5777 023574  012705  000010                  MOV     #8.,R5                  ;INIT COUNTER
 5778 023600  012701  024024                  MOV     #T122B,R1               ;SETUP POINTER TO DATA
 5779 023604  012737  030000  177776  1$:     MOV     #30000,@#177776         ;INIT PSW
 5780 023612  004767  000054                  JSR     PC,T122A                ; TEST INSTRUCTION
 5781 023616  022137  177776                  CMP     (R1)+,@#177776          ;IS PSW CORRECT
 5782 023622  001401                          BEQ     2$                      ;YES GO ON
 5783 023624  104001                          ERROR   +1              ;CPU ERROR
 5784                                                                 ;NO GO TO ERROR
 5785 023626  077512                  2$:     SOB     R5,1$                   ;REPEAT UNTIL ALL CASES ARE TESTED
 5786
 5787
 5788 023630  012705  000010                  MOV     #8.,R5                  ;INIT COUNTER
 5789 023634  012737  140000  177776  3$:     MOV     #140000,@#177776        ;SETUP PSW TO USER MODE
 5790 023642  012706  000600                  MOV     #600,R6                 ;SETUP USER STACK
 5791 023646  004767  000020                  JSR     PC,T122A                ; TEST INSTRUCTION
 5792 023652  022737  140017  177776          CMP     #140017,@#177776        ;IS PSW CORRECT
 5793 023660  001401                          BEQ     4$                      ;YES GO ON
 5794 023662  104001                          ERROR   +1              ;CPU ERROR
 5795                                                                 ;NO GO TO ERROR
 5796 023664  077515                  4$:     SOB     R5,3$                   ;REPEAT UNTIL ALL CASES ARE TESTED
 5797
 5798
 5799 023666  000167  000152                  JMP     FIN122
 5800                                 ;
 5801 023672  020527  000010          T122A:  CMP     R5,#8.                  ;FIND OUT WHAT COUNTER IS
 5802 023676  001003                          BNE     1$                      ;IF NOT PRIORITY 0 GO TO 1$
 5803 023700  000277                          SCC                             ;SET ALL CC BITS
 5804 023702  000230                          SPL     0                       ;SET PRIORITY TO 0
 5805 023704  000446                          BR      8$                      ;RETURN
 5806 023706  020527  000007          1$:     CMP     R5,#7                   ;FIND OUT WHAT COUNTER IS
 5807 023712  001003                          BNE     2$                      ;IF NOT PRIORITY 1 GO TO 2$
 5808 023714  000277                          SCC                             ;SET ALL CC BITS
 5809 023716  000231                          SPL     1                       ;SET PRIORITY TO 1
 5810 023720  000440                          BR      8$                      ;RETURN
 5811 023722  020527  000006          2$:     CMP     R5,#6                   ;FIND OUT WHAT COUNTER IS
 5812 023726  001003                          BNE     3$                      ;IF NOT PRIORITY 2 GO TO 3$
 5813 023730  000277                          SCC                             ;SET ALL CC BITS
 5814 023732  000232                          SPL     2                       ;SET PRIORITY TO 2
 5815 023734  000432                          BR      8$                      ;RETURN
 5816 023736  020527  000005          3$:     CMP     R5,#5                   ;FIND OUT WHAT COUNTER IS
 5817 023742  001003                          BNE     4$                      ;IF NOT PRIORITY 3 GO TO 4$
 5818 023744  000277                          SCC                             ;SET ALL CC BITS
 5819 023746  000233                          SPL     3                       ;SET PRIORITY TO 3
 5820 023750  000424                          BR      8$                      ;RETURN
 5821 023752  020527  000004          4$:     CMP     R5,#4                   ;FIND OUT WHAT COUNTER IS
 5822 023756  001003                          BNE     5$                      ;IF NOT PRIORITY 4 GO TO 5$
 5823 023760  000277                          SCC                             ;SET ALL CC BITS
 5824 023762  000234                          SPL     4                       ;SET PRIORITY TO 4
 5825 023764  000416                          BR      8$                      ;RETURN
 5826 023766  020527  000003          5$:     CMP     R5,#3                   ;FIND OUT WHAT COUNTER IS
 5827 023772  001003                          BNE     6$                      ;IF NOT PRIORITY 5 GO TO 6$
 5828 023774  000277                          SCC                             ;SET ALL CC BITS
```

```
********** DOUBLE OPERAND TESTS *****************

 5829 023776  000235                          SPL     5                       ;SET PRIORITY TO 5
 5830 024000  000410                          BR      8$                      ;RETURN
 5831 024002  020527  000002          6$:     CMP     R5,#2                   ;FIND OUT WHAT COUNTER IS
 5832 024006  001003                          BNE     7$                      ;IF NOT PRIORITY 6 GO TO 7$
 5833 024010  000277                          SCC                             ;SET ALL CC BITS
 5834 024012  000236                          SPL     6                       ;SET PRIORITY TO 6
 5835 024014  000402                          BR      8$                      ;RETURN
 5836 024016  000277                  7$:     SCC                             ;SET ALL CC BITS
 5837 024020  000237                          SPL     7                       ;SET PRIORITY TO 7
 5838 024022  000207                  8$:     RTS     PC                      ;RETURN
 5839                                  ;
 5840 024024  030017                  T122B:  .WORD   30017
 5841 024026  030057                          .WORD   30057
 5842 024030  030117                          .WORD   30117
 5843 024032  030157                          .WORD   30157
 5844 024034  030217                          .WORD   30217
 5845 024036  030257                          .WORD   30257
 5846 024040  030317                          .WORD   30317
 5847 024042  030357                          .WORD   30357
 5848 024044  000240                  FIN122: NOP
 5849                                  ;
 5852 024046                          TE123:
 5853                                  ;       TEST TSTSET INSTRUCTION (MULTI PROCESSING INST)
 5854 024046  005037  177776                  CLR     @#177776                ;INIT PSW
 5855 024052  012703  000012                  MOV     #10.,R3                 ;INIT COUNTER
 5856 024056  012701  000400                  MOV     #400,R1                 ;SETUP DESTINATION
 5857 024062  012700  024230                  MOV     #T123A,R0               ;SETUP SOURCE
 5858 024066  012021                  100$:   MOV     (R0)+,(R1)+             ;RELOCATE TABLES
 5859 024070  077302                          SOB     R3,100$                 ;ARE WE DONE
 5860 024072  013746  000010                  MOV     @#10,-(SP)              ;SAVE VECTOR
 5861 024076  012737  024254  000010          MOV     #T123D,@#10             ;SETUP NEW VECTOR
 5862 024104  005000                          CLR     R0                      ;INIT R0
 5863 024106  012701  000400                  MOV     #400,R1                 ;SETUP POINTERS TO TABLES
 5864 024112  012702  000410                  MOV     #410,R2                 ;
 5865 024116  012703  000416                  MOV     #416,R3                 ;
 5866 024122  010104                          MOV     R1,R4                   ;
 5867 024124  012737  030000  177776  1$:     MOV     #30000,@#177776         ;SETUP PSW
 5868 024132  000262                          SEV                             ;SET V BIT
 5869 024134  007221                          .WORD   7221                    ; TEST INSTRUCTION
 5870 024136  022237  177776                  CMP     (R2)+,@#177776          ;IS PSW CORRECT
 5871 024142  001401                          BEQ     2$                      ;YES GO ON
 5872 024144  104001                          ERROR   +1                      ;CPU ERROR
 5873                                                                         ;NO GO TO ERROR
 5874 024146  020013                  2$:     CMP     R0,(R3)                 ;IS R0 CORRECT
 5875 024150  001401                          BEQ     3$                      ;YES GO ON
 5876 024152  104001                          ERROR   +1                      ;CPU ERROR
 5877                                                                         ;NO GO TO ERROR
 5878 024154  005204                  3$:     INC     R4                      ;SETUP EXPECTED DATA
 5879 024156  005204                          INC     R4                      ;
 5880 024160  020401                          CMP     R4,R1                   ;IS R1 CORRECT
 5881 024162  001401                          BEQ     4$                      ;YES GO ON
 5882 024164  104001                          ERROR   +1                      ;CPU ERROR
 5883                                                                         ;NO GO TO ERROR
 5884 024166  052713  000001          4$:     BIS     #1,(R3)                 ;SETUP EXPECTED DATA
 5885 024172  022341                          CMP     (R3)+,-(R1)             ;IS TEST LOCATION CORRECT
 5886 024174  001401                          BEQ     5$                      ;YES GO ON
 5887 024176  104001                          ERROR   +1                      ;CPU ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

5888                                                            ;NO GO TO ERROR
5889 024200  005201                    5$:     INC     R1               ;POINT TO NEXT TEST LOCATION
5890 024202  005201                            INC     R1               ;
5891 024204  021127  177777                    CMP     (R1),#177777     ;ARE WE DONE
5892 024210  001345                            BNE     1$               ;NO GO TO 1$
5893 024212  012737  024256  000010            MOV     #T123E,@#10      ;SETUP NEW VECTOR
5894 024220  007201                            .WORD   7201             ; TEST INSTRUCTION ILLEGAL MODE
5895 024222  104001                            ERROR   .+1              ;CPU ERROR
5896                                                            ;GO TO ERROR IF DIDN'T TRAP
5897 024224  000167  000032                    JMP     T123F
5898                                    ;
5899                                    ;
5900 024230  167604                    T123A:  .WORD   167604
5901 024232  000000                            .WORD   0
5902 024234  000001                            .WORD   1
5903 024236  177777                            .WORD   177777
5904 024240  030010                    T123B:  .WORD   30010
5905 024242  030004                            .WORD   30004
5906 024244  030001                            .WORD   30001
5907 024246  167604                    T123C:  .WORD   167604
5908 024250  000000                            .WORD   0
5909 024252  000001                            .WORD   1
5910 024254  104001                    T123D:  ERROR   .+1              ;CPU ERROR
5911                                                            ;GO TO ERROR IF TRAPPED
5912 024256  005726                    T123E:  TST     (SP)+            ;CLEAN UP STACK
5913 024260  005726                            TST     (SP)+            ;
5914 024262  012637  000010            T123F:  MOV     (SP)+,@#10       ;RESTORE VECTOR
5915
5916                                    ;
5919 024266                            TE124:
5920                                    ;       TEST WRTLCK (WRITE LOCK MULTI PROCESSING INST)
5921 024266  005037  177776                    CLR     @#177776         ;INIT PSW
5922 024272  012703  000012                    MOV     #10.,R3          ;INIT COUNTER
5923 024276  012701  000400                    MOV     #400,R1          ;SETUP DESTINATION
5924 024302  012700  024462                    MOV     #T124A,R0        ;SETUP SOURCE
5925 024306  012021                    100$:   MOV     (R0)+,(R1)+      ;RELOCATE TABLES
5926 024310  077302                            SOB     R3,100$          ;ARE WE DONE
5927 024312  013746  000010                    MOV     @#10,-(SP)       ;SAVE VECTOR
5928 024316  012737  024506  000010            MOV     #T124D,@#10      ;SETUP NEW VECTOR
5929 024324  012701  000400                    MOV     #400,R1          ;SETUP POINTERS TO TABLES
5930 024330  012702  000410                    MOV     #410,R2          ;
5931 024334  012703  000416                    MOV     #416,R3          ;
5932 024340  010204                            MOV     R2,R4            ;
5933 024342  012737  030000  177776    1$:     MOV     #30000,@#177776  ;SETUP PSW
5934 024350  011100                            MOV     (R1),R0          ;SETUP R0
5935 024352  020327  000416                    CMP     R3,#416          ;IS THIS THE FIRST TEST CASE
5936 024356  001401                            BEQ     2$               ;YES GO TO 2$
5937 024360  000402                            BR      3$               ;NO GO TO 3$
5938 024362  000261                    2$:     SEC                      ;SET C BIT
5939 024364  000401                            BR      4$               ;
5940 024366  000241                    3$:     CLC                      ;CLEAR C BIT
5941 024370  000262                    4$:     SEV                      ;SET V BIT
5942 024372  007322                            .WORD   7322             ; TEST INSTRUCTION
5943 024374  022337  177776                    CMP     (R3)+,@#177776   ;IS PSW CORRECT
5944 024400  001401                            BEQ     5$               ;YES GO ON
5945 024402  104001                            ERROR   .+1              ;CPU ERROR
5946                                                            ;NO GO TO ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

 5947 024404  021100                 5$:     CMP     (R1),R0          ;IS R0 CORRECT
 5948 024406  001401                         BEQ     6$               ;YES GO ON
 5949 024410  104001                         ERROR   +1           ;CPU ERROR
 5950                                                             ;NO GO TO ERROR
 5951 024412  005204                 6$:     INC     R4               ;SETUP EXPECTED DATA
 5952 024414  005204                         INC     R4               ;
 5953 024416  020204                         CMP     R2,R4            ;IS R2 CORRECT
 5954 024420  001401                         BEQ     7$               ;YES GO ON
 5955 024422  104001                         ERROR   +1           ;CPU ERROR
 5956                                                             ;NO GO TO ERROR
 5957 024424  022142                 7$:     CMP     (R1)+,-(R2)      ;IS TEST LOCATION CORRECT
 5958 024426  001401                         BEQ     8$               ;YES GO ON
 5959 024430  104001                         ERROR   +1           ;CPU ERROR
 5960                                                             ;NO GO TO ERROR
 5961 024432  005202                 8$:     INC     R2               ;POINT TO NEXT TEST LOCATION
 5962 024434  005202                         INC     R2               ;
 5963 024436  021127  177777                 CMP     (R1),#177777     ;ARE WE DONE
 5964 024442  001337                         BNE     1$               ;NO GO TO 1$
 5965 024444  012737  024510  000010         MOV     #T124E,@#10      ;SETUP NEW VECTOR
 5966 024452  007302                         .WORD   7302             ; TEST INSTRUCTION ILLEGAL MODE
 5967 024454  104001                         ERROR   +1           ;CPU ERROR
 5968                                                             ;GO TO ERROR IF DIDN'T TRAP
 5969 024456  000167  000032                 JMP     T124F
 5970                                 ;
 5971                                 ;
 5972 024462  167604                 T124A:  .WORD   167604
 5973 024464  000000                         .WORD   0
 5974 024466  000001                         .WORD   1
 5975 024470  177777                         .WORD   177777
 5976 024472  177777                 T124B:  .WORD   177777
 5977 024474  177777                         .WORD   177777
 5978 024476  177777                         .WORD   177777
 5979 024500  030011                 T124C:  .WORD   30011
 5980 024502  030004                         .WORD   30004
 5981 024504  030000                         .WORD   30000
 5982 024506  104001                 T124D:  ERROR   +1               ;CPU ERROR
 5983                                                             ;GO TO ERROR IF TRAPPED
 5984 024510  005726                 T124E:  TST     (SP)+                ;CLEAN UP STACK
 5985 024512  005726                         TST     (SP)+                ;
 5986 024514  012637  000010         T124F:  MOV     (SP)+,@#10           ;RESTORE VECTOR
 5987
 5988                                 ;
 5991 024520                         TE125:
 5992                                 ;       TEST MUL (MULTIPLY INST)
 5993 024520  005037  177776                 CLR     @#177776             ;INIT PS
 5994 024524  012701  024760                 MOV     #TE125A,R1           ;SETUP POINTERS TO TABLES
 5995
 5996 024530  010137  114146         1$:     MOV     R1,@#EXPDAT          ;
 5997 024534  062737  000002  114146         ADD     #2,@#EXPDAT          ;POINT TO SOURCE
 5998 024542  012703  122222                 MOV     #122222,R3           ;INIT R3 TO A KNOWN STATE
 5999 024546  011102                         MOV     (R1),R2          ;INIT DESTINATION REG
 6000 024550  000277                         SCC                      ;SET ALL CC BITS
 6001 024552  070261  000002                 MUL     2(R1),R2             ; TEST INSTRUCTION
 6002 024556  026137  000004  177776         CMP     4(R1),@#177776       ;IS PS CORRECT
 6003 024564  001401                         BEQ     2$               ;YES GO ON
 6004 024566  104001                         ERROR   +1               ;CPU ERROR
 6005                                                                 ;NO GO TO ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

 6006 024570  026103  000006          2$:     CMP     6(R1),R3                ;IS R3 CORRECT
 6007 024574  001401                          BEQ     3$                      ;YES GO ON
 6008 024576  104001                          ERROR   .+1                  ;CPU ERROR
 6009                                                                      ;NO GO TO ERROR
 6010 024600  026102  000010          3$:     CMP     10(R1),R2               ;IS R2 CORRECT
 6011 024604  001401                          BEQ     4$                      ;YES GO ON
 6012 024606  104001                          ERROR   .+1                  ;CPU ERROR
 6013                                                                      ;NO GO TO ERROR
 6014 024610  026177  000002  067330  4$:     CMP     2(R1),@EXPDAT           ;IS SOURCE LOCATION OK
 6015 024616  001401                          BEQ     5$                      ;YES GO ON
 6016 024620  104001                          ERROR   .+1                  ;CPU ERROR
 6017                                                                      ;NO GO TO ERROR
 6018 024622  062701  000012          5$:     ADD     #12,R1                  ;GO TO NEXT TEST
 6019 024626  020127  025206                  CMP     R1,#FIN125              ;ARE WE FINISHED
 6020 024632  001336                          BNE     1$                      ;NO GO TO 1$
 6021
 6022                                 ;
 6023                                 ;SECOND PART
 6024                                 ;USING ODD REGISTER
 6025                                 ;
 6026 024634  012701  024760          6$:     MOV     #TE125A,R1              ;SETUP POINTERS TO TABLES
 6027 024640                          7$:
 6028 024640  010102                          MOV     R1,R2                   ;
 6029 024642  012706  001000                  MOV     #STBOT,R6               ;INIT R6 TO A KNOWN STATE
 6030 024646  012704  000004                  MOV     #4,R4                   ;SETUP R4 VALUE
 6031 024652  011105                          MOV     (R1),R5             ;INIT DESTINATION REG
 6032 024654  000277                          SCC                             ;SET ALL CC BITS
 6033 024656  070561  000002                  MUL     2(R1),R5                ; TEST INSTRUCTION
 6034 024662  026137  000004  177776          CMP     4(R1),@#177776          ;IS PS CORRECT
 6035 024670  001401                          BEQ     8$                      ;YES GO ON
 6036 024672  104001                          ERROR   .+1                  ;CPU ERROR
 6037                                                                      ;NO GO TO ERROR
 6038 024674  026105  000006          8$:     CMP     6(R1),R5                ;IS R5 CORRECT
 6039 024700  001401                          BEQ     9$                      ;YES GO ON
 6040 024702  104001                          ERROR   .+1                  ;CPU ERROR
 6041                                                                      ;NO GO TO ERROR
 6042 024704  020627  001000          9$:     CMP     R6,#STBOT               ;IS R6 CORRECT
 6043 024710  001403                          BEQ     10$                     ;YES GO ON
 6044 024712  012706  001000                  MOV     #STBOT,R6               ;RESTORE SP
 6045 024716  104001                          ERROR   .+1                  ;CPU ERROR
 6046                                                                      ;NO GO TO ERROR
 6047 024720  005722                  10$:    TST     (R2)+                   ;POINT TO SOURCE OPERAND
 6048 024722  021261  000002                  CMP     (R2),2(R1)              ;IS SOURCE LOCATION OK
 6049 024726  001401                          BEQ     11$                     ;YES GO ON
 6050 024730  104001                          ERROR   .+1                  ;CPU ERROR
 6051                                                                      ;NO GO TO ERROR
 6052 024732  020427  000004          11$:    CMP     R4,#4                   ;IS R4 CORRECT
 6053 024736  001401                          BEQ     12$                     ;YES GO ON
 6054 024740  104001                          ERROR   .+1                  ;CPU ERROR
 6055                                                                      ;NO GO TO ERROR
 6056 024742  062701  000012          12$:    ADD     #12,R1                  ;
 6057 024746  020127  025206                  CMP     R1,#FIN125              ;ARE WE FINISHED
 6058 024752  001332                          BNE     7$                      ;NO GO TO 7$
 6059
 6060
 6061 024754  000167  000226                  JMP     FIN125
 6062                                 ;
```

********** DOUBLE OPERAND TESTS **********

```
6063
6064 024760  177777                ;
6065 024762  177777        TE125A: .WORD   177777          ;MULTIPLICAND
6066 024764  000000                .WORD   177777          ;MULTIPLIER
6067 024766  000001                .WORD   0
6068 024770  000000                .WORD   1
6069                               .WORD   0
6070 024772  006772
6071 024774  100000                .WORD   6772            ;MULTIPLICAND
6072 024776  000011                .WORD   100000          ;MULTIPLIER
6073 025000  000000                .WORD   11
6074 025002  174403                .WORD   0
6075                               .WORD   174403
6076 025004  177777
6077 025006  077777                .WORD   177777          ;MULTIPLICAND
6078 025010  000010                .WORD   77777           ;MULTIPLIER
6079 025012  100001                .WORD   10
6080 025014  177777                .WORD   100001
6081                               .WORD   177777
6082 025016  077777
6083 025020  000456                .WORD   77777           ;MULTIPLICAND
6084 025022  000001                .WORD   456             ;MULTIPLIER
6085 025024  177322                .WORD   1
6086 025026  000226                .WORD   177322
6087                               .WORD   226
6088 025030  173210
6089 025032  000000                .WORD   173210          ;MULTIPLICAND
6090 025034  000004                .WORD   0               ;MULTIPLIER
6091 025036  000000                .WORD   4
6092 025040  000000                .WORD   0
6093                               .WORD   0
6094 025042  000000
6095 025044  003251                .WORD   0               ;MULTIPLICAND
6096 025046  000004                .WORD   3251            ;MULTIPLIER
6097 025050  000000                .WORD   4
6098 025052  000000                .WORD   0
6099                               .WORD   0
6100 025054  000000
6101 025056  000000                .WORD   0               ;MULTIPLICAND
6102 025060  000004                .WORD   0               ;MULTIPLIER
6103 025062  000000                .WORD   4
6104 025064  000000                .WORD   0
6105                               .WORD   0
6106 025066  100000
6107 025070  000001                .WORD   100000          ;MULTIPLICAND
6108 025072  000010                .WORD   1               ;MULTIPLIER
6109 025074  100000                .WORD   10
6110 025076  177777                .WORD   100000
6111                               .WORD   177777
6112 025100  077777
6113 025102  000001                .WORD   77777           ;MULTIPLICAND
6114 025104  000000                .WORD   1               ;MULTIPLIER
6115 025106  077777                .WORD   0
6116 025110  000000                .WORD   77777
6117                               .WORD   0
6118 025112  000010
6119 025114  010000                .WORD   10              ;MULTIPLICAND
                                   .WORD   10000           ;MULTIPLIER
```

```
********** DOUBLE OPERAND TESTS *****************

 6120 025116  000001                          .WORD   1
 6121 025120  100000                          .WORD   100000
 6122 025122  000000                          .WORD   0
 6123
 6124 025124  001452                          .WORD   1452                    ;MULTIPLICAND
 6125 025126  034527                          .WORD   34527                   ;MULTIPLIER
 6126 025130  000001                          .WORD   1
 6127 025132  066506                          .WORD   66506
 6128 025134  000265                          .WORD   265
 6129
 6130 025136  000007                          .WORD   7                       ;MULTIPLICAND
 6131 025140  000400                          .WORD   400                     ;MULTIPLIER
 6132 025142  000000                          .WORD   0
 6133 025144  003400                          .WORD   3400
 6134 025146  000000                          .WORD   0
 6135
 6136 025150  000002                          .WORD   2                       ;MULTIPLICAND
 6137 025152  100000                          .WORD   100000                  ;MULTIPLIER
 6138 025154  000011                          .WORD   11
 6139 025156  000000                          .WORD   0
 6140 025160  177777                          .WORD   177777
 6141
 6142 025162  100000                          .WORD   100000                  ;MULTIPLICAND
 6143 025164  077777                          .WORD   77777                   ;MULTIPLIER
 6144 025166  000011                          .WORD   11
 6145 025170  100000                          .WORD   100000
 6146 025172  140000                          .WORD   140000
 6147
 6148 025174  000001                          .WORD   1                       ;MULTIPLICAND
 6149 025176  177777                          .WORD   177777                  ;MULTIPLIER
 6150 025200  000010                          .WORD   10
 6151 025202  177777                          .WORD   177777
 6152 025204  177777                          .WORD   177777
 6153 025206                          FIN125:
 6154                                 ;
 6157 025206                          TE126:
 6158                                 ;       TEST DIV (DIVIDE INST)
 6159 025206  005037  177776                  CLR     @#177776                ;INIT PSW
 6160 025212  005006                          CLR     R6                      ;INIT SP
 6161 025214  013705  000000                  MOV     @#0,R5                  ;SAVE VECTORS
 6162 025220  013701  000002                  MOV     @#2,R1                  ;
 6163 025224  012737  000137  000000          MOV     #137,@#0                ;SETUP NEW VECTORS
 6164 025232  012737  025254  000002          MOV     #TE126A,@#2             ;
 6165 025240  000277                          SCC                             ;SET ALL CC BITS
 6166 025242  071627  000002                  DIV     #2,R6                   ; TEST INSTRUCTION
 6167 025246  012706  001000          A126:   MOV     #STBOT,R6               ;RESTORE SP BEFORE GOING TO ERROR
 6168 025252  104001                          ERROR   +1                      ;CPU ERROR
 6169                                                                         ;IF R7 ISN'T CORRECT GO TO ERROR
 6170 025254  022737  000000  177776  TE126A: CMP     #0,@#177776             ;IS PS CORRECT
 6171 025262  001403                          BEQ     1$                      ;YES GO ON
 6172 025264  012706  001000                  MOV     #STBOT,R6               ;RESTORE SP BEFORE GOING TO ERROR
 6173 025270  104001                          ERROR   +1                      ;CPU ERROR
 6174                                                                         ;NO GO TO ERROR
 6175 025272  012704  025246          1$:     MOV     #A126,R4                ;SETUP EXPECTED DATA
 6176 025276  006204                          ASR     R4                      ;
 6177 025300  020406                          CMP     R4,R6                   ;IS R6 CORRECT
 6178 025302  001403                          BEQ     2$                      ;YES GO ON
```

********** DOUBLE OPERAND TESTS **********

```
6179 025304  012706  001000                  MOV     #STBOT,R6           ;RESTORE SP BEFORE GOING TO ERROR
6180 025310  104001                          ERROR   ⋅1                  ;CPU ERROR
6181                                                                     ;NO GO TO ERROR
6182 025312  010537  000000          2$:     MOV     R5,@#0              ;RESTORE VECTORS
6183 025316  010137  000002                  MOV     R1,@#2              ;
6184 025322  012706  001000                  MOV     #STBOT,R6           ;INIT SP
6185 025326  012702  000006                  MOV     #6,R2               ;INIT GPR 2
6186 025332  012703  000047                  MOV     #47,R3              ;INIT GPR 3
6187 025336  000277                          SCC                         ;SET ALL CC BITS
6188 025340  071302                          DIV     R2,R3               ; TEST INSTRUCTION
6189 025342  022737  000002  177776          CMP     #2,@#177776         ;IS PS CORRECT
6190 025350  001401                          BEQ     3$                  ;YES GO ON
6191 025352  104001                          ERROR   ⋅1                  ;CPU ERROR
6192                                                                     ;NO GO TO ERROR
6193 025354  022702  000006          3$:     CMP     #6,R2               ;IS R2 CORRECT
6194 025360  001401                          BEQ     4$                  ;YES GO ON
6195 025362  104001                          ERROR   ⋅1                  ;CPU ERROR
6196                                                                     ;NO GO TO ERROR
6197 025364  022703  000047          4$:     CMP     #47,R3              ;IS R3 CORRECT
6198 025370  001401                          BEQ     5$                  ;YES GO ON
6199 025372  104001                          ERROR   ⋅1                  ;CPU ERROR
6200                                                                     ;NO GO TO ERROR
6201 025374  005004                  5$:     CLR     R4                  ;INIT R4
6202 025376  012705  000004                  MOV     #4,R5               ;INIT R5
6203 025402  000277                          SCC                         ;SET ALL CC BITS
6204 025404  071427  000000                  DIV     #0,R4               ; TEST INSTRUCTION
6205 025410  022737  000007  177776          CMP     #7,@#177776         ;IS PS CORRECT
6206 025416  001401                          BEQ     6$                  ;YES GO ON
6207 025420  104001                          ERROR   ⋅1                  ;CPU ERROR
6208                                                                     ;NO GO TO ERROR
6209 025422  022704  000000          6$:     CMP     #0,R4               ;IS R4 CORRECT
6210 025426  001401                          BEQ     7$                  ;YES GO ON
6211 025430  104001                          ERROR   ⋅1                  ;CPU ERROR
6212                                                                     ;NO GO TO ERROR
6213 025432  022705  000004          7$:     CMP     #4,R5               ;IS R5 CORRECT
6214 025436  001401                          BEQ     8$                  ;YES GO ON
6215 025440  104001                          ERROR   ⋅1                  ;CPU ERROR
6216                                                                     ;NO GO TO ERROR
6217 025442  012700  000004          8$:     MOV     #4,R0               ;INIT R0
6218 025446  012705  000010                  MOV     #10,R5              ;INIT R5
6219 025452  005004                          CLR     R4                  ;INIT R4
6220 025454  000277                          SCC                         ;SET ALL CC BITS
6221 025456  071400                          DIV     R0,R4               ; TEST INSTRUCTION
6222 025460  022737  000000  177776          CMP     #0,@#177776         ;IS PS CORRECT
6223 025466  001405                          BEQ     9$                  ;YES GO ON
6224 025470  010067  152704                  MOV     R0,400              ;SAVE R0
6225 025474  104001                          ERROR   ⋅1                  ;CPU ERROR
6226                                                                     ;NO GO TO ERROR
6227 025476  016700  152676                  MOV     400,R0              ;RESTORE R0
6228 025502  022700  000004          9$:     CMP     #4,R0               ;IS R0 CORRECT
6229 025506  001401                          BEQ     10$                 ;YES GO ON
6230 025510  104001                          ERROR   ⋅1                  ;CPU ERROR
6231                                                                     ;NO GO TO ERROR
6232 025512  022704  000002          10$:    CMP     #2,R4               ;IS R4 CORRECT
6233 025516  001401                          BEQ     11$                 ;YES GO ON
6234 025520  104001                          ERROR   ⋅1                  ;CPU ERROR
6235                                                                     ;NO GO TO ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

 6236 025522  022705  000000          11$:    CMP     #0,R5                   ;IS R5 CORRECT
 6237 025526  001401                          BEQ     12$                     ;YES GO ON
 6238 025530  104001                          ERROR   +1              ;CPU ERROR
 6239                                                                 ;NO GO TO ERROR
 6240 025532  012705  000010          12$:    MOV     #10,R5                  ;INIT R5
 6241 025536  005004                          CLR     R4                      ;INIT R4
 6242 025540  000277                          SCC                             ;SET ALL CC BITS
 6243 025542  071427  000003                  DIV     #3,R4                   ; TEST INSTRUCTION
 6244 025546  022737  000000  177776          CMP     #0,@#177776             ;IS PS CORRECT
 6245 025554  001401                          BEQ     13$                     ;YES GO ON
 6246 025556  104001                          ERROR   +1              ;CPU ERROR
 6247                                                                 ;NO GO TO ERROR
 6248 025560  022704  000002          13$:    CMP     #2,R4                   ;IS R4 CORRECT
 6249 025564  001401                          BEQ     14$                     ;YES GO ON
 6250 025566  104001                          ERROR   +1              ;CPU ERROR
 6251                                                                 ;NO GO TO ERROR
 6252 025570  022705  000002          14$:    CMP     #2,R5                   ;IS R5 CORRECT
 6253 025574  001401                          BEQ     15$                     ;YES GO ON
 6254 025576  104001                          ERROR   +1              ;CPU ERROR
 6255                                                                 ;NO GO TO ERROR
 6256                                  ;
 6257                                  ;
 6258                                  ;
 6259 025600  012701  025730          15$:    MOV     #TE126B,R1              ;SETUP POINTERS TO TABLES
 6260
 6261 025604  010137  114146          16$:    MOV     R1,@#EXPDAT             ;SAVE A COPY OF R1
 6262 025610  011104                          MOV     (R1),R4         ;INIT R4
 6263 025612  016103  000004                  MOV     4(R1),R3                ;SAVE SOURCE
 6264                                  ;
 6265 025616  016105  000002                  MOV     2(R1),R5                ;INIT R5
 6266 025622  000277                          SCC                             ;SET ALL CC BITS
 6267 025624  071461  000004                  DIV     4(R1),R4                ; TEST INSTRUCTION
 6268 025630  026137  000006  177776          CMP     6(R1),@#177776          ;IS PS CORRECT
 6269 025636  001401                          BEQ     17$                     ;YES GO ON
 6270 025640  104001                          ERROR   +1              ;CPU ERROR
 6271                                                                 ;NO GO TO ERROR
 6272 025642  026105  000010          17$:    CMP     10(R1),R5               ;IS R5 CORRECT
 6273 025646  001401                          BEQ     18$                     ;YES GO ON
 6274 025650  104001                          ERROR   +1              ;CPU ERROR
 6275                                                                 ;NO GO TO ERROR
 6276 025652  026104  000012          18$:    CMP     12(R1),R4               ;IS R4 CORRECT
 6277 025656  001401                          BEQ     19$                     ;YES GO ON
 6278 025660  104001                          ERROR   +1              ;CPU ERROR
 6279                                                                 ;NO GO TO ERROR
 6280 025662  023701  114146          19$:    CMP     @#EXPDAT,R1             ;IS R1 CORRECT
 6281 025666  001403                          BEQ     20$                     ;YES GO ON
 6282 025670  104001                          ERROR   +1              ;CPU ERROR
 6283                                                                 ;NO GO TO ERROR
 6284 025672  013701  114146                  MOV     @#EXPDAT,R1             ;RESTORE CORRECT VALUE
 6285 025676  026103  000004          20$:    CMP     4(R1),R3                ;IS SOURCE CORRECT
 6286 025702  001403                          BEQ     21$                     ;YES GO ON
 6287 025704  104001                          ERROR   +1              ;CPU ERROR
 6288                                                                 ;NO GO TO ERROR
 6289 025706  010361  000004                  MOV     R3,4(R1)                        ;TRY TO RESTORE CODE
 6290 025712  062701  000014          21$:    ADD     #14,R1                  ;POINT TO NEXT LOCATION
 6291 025716  021127  000333                  CMP     (R1),#333               ;ARE WE DONE
 6292 025722  001330                          BNE     16$                     ;NO GO TO 16$
```

```
********** DOUBLE OPERAND TESTS ******************

                                                    JMP     FIN126
                                                ;
                                                ;
                                                TE126B: .WORD   177777  ;DIVIDEND
                                                        .WORD   177777  ;INIT R5
                                                        .WORD   177777  ;DIVISOR
                                                        .WORD   0       ;PSW
                                                        .WORD   0       ;R5 RESULT
                                                        .WORD   1       ;R4 RESULT

                                                        .WORD   0       ;DIVIDEND
                                                        .WORD   177777  ;INIT R5
                                                        .WORD   177777  ;DIVISOR
                                                        .WORD   12      ;PSW
                                                        .WORD   177777  ;R5 RESULT
                                                        .WORD   0       ;R4 RESULT

                                                        .WORD   177777  ;DIVIDEND
                                                        .WORD   0       ;INIT R5
                                                        .WORD   177777  ;DIVISOR
                                                        .WORD   2       ;PSW
                                                        .WORD   0       ;R5 RESULT
                                                        .WORD   177777  ;R4 RESULT

                                                        .WORD   0       ;DIVIDEND
                                                        .WORD   7642    ;INIT R5
                                                        .WORD   7643    ;DIVISOR
                                                        .WORD   4       ;PSW
                                                        .WORD   7642    ;R5 RESULT
                                                        .WORD   0       ;R4 RESULT

                                                        .WORD   0       ;DIVIDEND
                                                        .WORD   137     ;INIT R5
                                                        .WORD   177543  ;DIVISOR
                                                        .WORD   4       ;PSW
                                                        .WORD   137     ;R5 RESULT
                                                        .WORD   0       ;R4 RESULT

                                                        .WORD   0       ;DIVIDEND
                                                        .WORD   7643    ;INIT R5
                                                        .WORD   7643    ;DIVISOR
                                                        .WORD   0       ;PSW
                                                        .WORD   0       ;R5 RESULT
                                                        .WORD   1       ;R4 RESULT

                                                        .WORD   100000  ;DIVIDEND
                                                        .WORD   4376    ;INIT R5
                                                        .WORD   10021   ;DIVISOR
                                                        .WORD   12      ;PSW
                                                        .WORD   4376    ;R5 RESULT
                                                        .WORD   100000  ;R4 RESULT

                                                        .WORD   177700  ;DIVIDEND
                                                        .WORD   170033  ;INIT R5
                                                        .WORD   10021   ;DIVISOR
```

********** DOUBLE OPERAND TESTS **********

```
6350 026062  000010                    .WORD   10      ;PSW
6351 026064  171307                    .WORD   171307  ;R5 RESULT
6352 026066  176024                    .WORD   176024  ;R4 RESULT
6353
6354 026070  177700                    .WORD   177700  ;DIVIDEND
6355 026072  170033                    .WORD   170033  ;INIT R5
6356 026074  167757                    .WORD   167757  ;DIVISOR
6357 026076  000000                    .WORD   0       ;PSW
6358 026100  171307                    .WORD   171307  ;R5 RESULT
6359 026102  001754                    .WORD   1754    ;R4 RESULT
6360
6361 026104  000000                    .WORD   0       ;DIVIDEND
6362 026106  177777                    .WORD   177777  ;INIT R5
6363 026110  000001                    .WORD   1       ;DIVISOR
6364 026112  000002                    .WORD   2       ;PSW
6365 026114  177777                    .WORD   177777  ;R5 RESULT
6366 026116  000000                    .WORD   0       ;R4 RESULT
6367
6368 026120  177777                    .WORD   177777  ;DIVIDEND
6369 026122  045716                    .WORD   45716   ;INIT R5
6370 026124  000001                    .WORD   1       ;DIVISOR
6371 026126  000012                    .WORD   12      ;PSW
6372 026130  045716                    .WORD   45716   ;R5 RESULT
6373 026132  177777                    .WORD   177777  ;R4 RESULT
6374
6375 026134  000000                    .WORD   0       ;DIVIDEND
6376 026136  000002                    .WORD   2       ;INIT R5
6377 026140  177770                    .WORD   177770  ;DIVISOR
6378 026142  000004                    .WORD   4       ;PSW
6379 026144  000002                    .WORD   2       ;R5 RESULT
6380 026146  000000                    .WORD   0       ;R4 RESULT
6381
6382 026150  177777                    .WORD   177777  ;DIVIDEND
6383 026152  177776                    .WORD   177776  ;INIT R5
6384 026154  000010                    .WORD   10      ;DIVISOR
6385 026156  000004                    .WORD   4       ;PSW
6386 026160  177776                    .WORD   177776  ;R5 RESULT
6387 026162  000000                    .WORD   0       ;R4 RESULT
6388
6389 026164  000001                    .WORD   1       ;DIVIDEND
6390 026166  177777                    .WORD   177777  ;INIT R5
6391 026170  000001                    .WORD   1       ;DIVISOR
6392 026172  000002                    .WORD   2       ;PSW
6393 026174  177777                    .WORD   177777  ;R5 RESULT
6394 026176  000001                    .WORD   1       ;R4 RESULT
6395
6396 026200  000001                    .WORD   1       ;DIVIDEND
6397 026202  000000                    .WORD   0       ;INIT R5
6398 026204  000002                    .WORD   2       ;DIVISOR
6399 026206  000002                    .WORD   2       ;PSW
6400 026210  000000                    .WORD   0       ;R5 RESULT
6401 026212  000001                    .WORD   1       ;R4 RESULT
6402
6403 026214  000001                    .WORD   1       ;DIVIDEND
6404 026216  000000                    .WORD   0       ;INIT R5
6405 026220  000003                    .WORD   3       ;DIVISOR
6406 026222  000000                    .WORD   0       ;PSW
```

```
********** DOUBLE OPERAND TESTS *****************

6407 026224  000001                          .WORD   1       ;R5 RESULT
6408 026226  052525                          .WORD   52525   ;R4 RESULT
6409
6410 026230  000023                          .WORD   23      ;DIVIDEND
6411 026232  016054                          .WORD   16054   ;INIT R5
6412 026234  016537                          .WORD   16537   ;DIVISOR
6413 026236  000000                          .WORD   0       ;PSW
6414 026240  010222                          .WORD   10222   ;R5 RESULT
6415 026242  000246                          .WORD   246     ;R4 RESULT
6416
6417 026244  000333                          .WORD   333
6418 026246                          FIN126:
6419                                 ;
6422 026246                          TE127:
6423                                 ;       TEST ASH (ARITHMETIC SHIFT)
6424 026246  005037  177776                  CLR     @#177776                ;INIT PSW
6425 026252  012702  000001                  MOV     #1,R2                   ;SETUP OPERAND
6426 026256  000277                          SCC                             ;SET ALL CC BITS
6427 026260  072202                          ASH     R2,R2                   ; TEST INSTRUCTION
6428 026262  022737  000000  177776          CMP     #0,@#177776             ;IS PS CORRECT
6429 026270  001401                          BEQ     1$                      ;YES GO ON
6430 026272  104001                          ERROR   .+1             ;CPU ERROR
6431                                                                 ;NO GO TO ERROR
6432 026274  020227  000002          1$:     CMP     R2,#2                   ;IS R2 CORRECT
6433 026300  001401                          BEQ     2$                      ;YES GO ON
6434 026302  104001                          ERROR   .+1             ;CPU ERROR
6435                                                                 ;NO GO TO ERROR
6436 026304  012702  100000          2$:     MOV     #100000,R2              ;SETUP R2
6437 026310  012703  000001                  MOV     #1,R3                   ;SETUP R3
6438 026314  000257                          CCC                             ;CLEAR ALL CC BITS
6439 026316  072203                          ASH     R3,R2                   ; TEST INSTRUCTION
6440 026320  022737  000007  177776          CMP     #7,@#177776             ;IS PS CORRECT
6441 026326  001401                          BEQ     3$                      ;YES GO ON
6442 026330  104001                          ERROR   .+1             ;CPU ERROR
6443                                                                 ;NO GO TO ERROR
6444 026332  020327  000001          3$:     CMP     R3,#1                   ;IS R3 CORRECT
6445 026336  001401                          BEQ     4$                      ;YES GO ON
6446 026340  104001                          ERROR   .+1             ;CPU ERROR
6447                                                                 ;NO GO TO ERROR
6448 026342  020227  000000          4$:     CMP     R2,#0                   ;IS R2 CORRECT
6449 026346  001401                          BEQ     5$                      ;YES GO ON
6450 026350  104001                          ERROR   .+1             ;CPU ERROR
6451                                                                 ;NO GO TO ERROR
6452 026352  012701  026446          5$:     MOV     #TE127A,R1              ;SETUP POINTERS TO TABLES
6453
6454 026356  010103                  6$:     MOV     R1,R3                   ;
6455 026360  016102  000002                  MOV     2(R1),R2                ;SETUP R2
6456 026364  000277                          SCC                             ;SET ALL CC BITS
6457 026366  072211                          ASH     (R1),R2                 ; TEST INSTRUCTION
6458 026370  026137  000004  177776          CMP     4(R1),@#177776          ;IS PS CORRECT
6459 026376  001401                          BEQ     7$                      ;YES GO ON
6460 026400  104001                          ERROR   .+1             ;CPU ERROR
6461                                                                 ;NO GO TO ERROR
6462 026402  026102  000006          7$:     CMP     6(R1),R2                ;IS R2 CORRECT
6463 026406  001401                          BEQ     8$                      ;YES GO ON
6464 026410  104001                          ERROR   .+1             ;CPU ERROR
6465                                                                 ;NO GO TO ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

 6466 026412  020301                8$:     CMP     R3,R1                   ;IS R1 CORRECT
 6467 026414  001402                        BEQ     9$                      ;YES GO ON
 6468 026416  104001                        ERROR   +1              ;CPU ERROR
 6469                                                               ;NO GO TO ERROR
 6470 026420  010301                        MOV     R3,R1                   ;RESTORE R1
 6471 026422  021311                9$:     CMP     (R3),(R1)               ;IS SOURCE CORRECT
 6472 026424  001401                        BEQ     10$                     ;YES GO ON
 6473 026426  104001                        ERROR   +1              ;CPU ERROR
 6474                                                               ;NO GO TO ERROR
 6475                                                               ;SOURCE LOOKS INCORRECT
 6476 026430  062701  000010       10$:    ADD     #10,R1                  ;INCREMENT POINTER
 6477 026434  020127  026706               CMP     R1,#FIN127              ;ARE WE DONE
 6478 026440  001346                        BNE     6$                      ;NO GO TO 6$
 6479
 6480
 6481 026442  000167  000240               JMP     FIN127
 6482                                ;
 6483                                ;
 6484 026446  177761                TE127A: .WORD   177761          ;SOURCE
 6485 026450  077777                        .WORD   77777           ;DEST
 6486 026452  000005                        .WORD   5
 6487 026454  000000                        .WORD   0
 6488
 6489 026456  177700                        .WORD   177700          ;SOURCE
 6490 026460  017777                        .WORD   17777           ;DEST
 6491 026462  000000                        .WORD   0
 6492 026464  017777                        .WORD   17777
 6493
 6494 026466  177700                        .WORD   177700          ;SOURCE
 6495 026470  100000                        .WORD   100000          ;DEST
 6496 026472  000010                        .WORD   10
 6497 026474  100000                        .WORD   100000
 6498
 6499 026476  177777                        .WORD   177777          ;SOURCE
 6500 026500  100000                        .WORD   100000          ;DEST
 6501 026502  000010                        .WORD   10
 6502 026504  140000                        .WORD   140000
 6503
 6504 026506  177737                        .WORD   177737          ;SOURCE
 6505 026510  177777                        .WORD   177777          ;DEST
 6506 026512  000011                        .WORD   11
 6507 026514  177777                        .WORD   177777
 6508
 6509 026516  177706                        .WORD   177706          ;SOURCE
 6510 026520  102000                        .WORD   102000          ;DEST
 6511 026522  000007                        .WORD   7
 6512 026524  000000                        .WORD   0
 6513
 6514 026526  177710                        .WORD   177710          ;SOURCE
 6515 026530  017777                        .WORD   17777           ;DEST
 6516 026532  000013                        .WORD   13
 6517 026534  177400                        .WORD   177400
 6518
 6519 026536  177713                        .WORD   177713          ;SOURCE
 6520 026540  000012                        .WORD   12              ;DEST
 6521 026542  000000                        .WORD   0
 6522 026544  050000                        .WORD   50000
```

********** DOUBLE OPERAND TESTS **********

```
6523
6524 026546  177707                    .WORD   177707          ;SOURCE
6525 026550  170001                    .WORD   170001          ;DEST
6526 026552  000002                    .WORD   2
6527 026554  000200                    .WORD   200
6528
6529 026556  177717                    .WORD   177717          ;SOURCE
6530 026560  000001                    .WORD   1               ;DEST
6531 026562  000012                    .WORD   12
6532 026564  100000                    .WORD   100000
6533
6534 026566  177740                    .WORD   177740          ;SOURCE
6535 026570  017777                    .WORD   17777           ;DEST
6536 026572  000004                    .WORD   4
6537 026574  000000                    .WORD   0
6538
6539 026576  177771                    .WORD   177771          ;SOURCE
6540 026600  150000                    .WORD   150000          ;DEST
6541 026602  000010                    .WORD   10
6542 026604  177640                    .WORD   177640
6543
6544 026606  177742                    .WORD   177742          ;SOURCE
6545 026610  100000                    .WORD   100000          ;DEST
6546 026612  000011                    .WORD   11
6547 026614  177777                    .WORD   177777
6548
6549 026616  177764                    .WORD   177764          ;SOURCE
6550 026620  100000                    .WORD   100000          ;DEST
6551 026622  000010                    .WORD   10
6552 026624  177770                    .WORD   177770
6553
6554 026626  177750                    .WORD   177750          ;SOURCE
6555 026630  052525                    .WORD   52525           ;DEST
6556 026632  000004                    .WORD   4
6557 026634  000000                    .WORD   0
6558
6559 026636  177760                    .WORD   177760          ;SOURCE
6560 026640  100000                    .WORD   100000          ;DEST
6561 026642  000011                    .WORD   11
6562 026644  177777                    .WORD   177777
6563
6564 026646  177770                    .WORD   177770          ;SOURCE
6565 026650  100000                    .WORD   100000          ;DEST
6566 026652  000010                    .WORD   10
6567 026654  177600                    .WORD   177600
6568
6569 026656  177712                    .WORD   177712          ;SOURCE
6570 026660  004367                    .WORD   4367            ;DEST
6571 026662  000013                    .WORD   13
6572 026664  156000                    .WORD   156000
6573
6574 026666  177764                    .WORD   177764          ;SOURCE
6575 026670  017777                    .WORD   17777           ;DEST
6576 026672  000001                    .WORD   1
6577 026674  000001                    .WORD   1
6578
6579 026676  177701                    .WORD   177701          ;SOURCE
```

********** DOUBLE OPERAND TESTS **********

```
6580 026700  110000                          .WORD   110000          ;DEST
6581 026702  000003                          .WORD   3
6582 026704  020000                          .WORD   20000
6583
6584 026706  000240                FIN127: NOP
6585                                       ;
6588 026710                        TE130:
6589                               ;       TEST ASHC (ARITHMETIC SHIFT COMBINED)
6590 026710  005037  177776                 CLR     @#177776                ;INIT PSW
6591 026714  012701  000023                 MOV     #23,R1                  ;SETUP R1
6592 026720  012705  052525                 MOV     #52525,R5               ;SETUP R5
6593 026724  005004                         CLR     R4                      ;SETUP R4
6594 026726  000277                         SCC                             ;SET ALL CC BITS
6595 026730  073401                         ASHC    R1,R4                   ; TEST INSTRUCTION
6596 026732  023727  177776  000012         CMP     @#177776,#12            ;IS PS CORRECT
6597 026740  001401                         BEQ     1$                      ;YES GO ON
6598 026742  104001                         ERROR   +1              ;CPU ERROR
6599                                                                 ;NO GO TO ERROR
6600 026744  020127  000023          1$:    CMP     R1,#23                  ;IS R1 CORRECT
6601 026750  001401                         BEQ     2$                      ;YES GO ON
6602 026752  104001                         ERROR   +1              ;CPU ERROR
6603                                                                 ;NO GO TO ERROR
6604 026754  020427  125250          2$:    CMP     R4,#125250              ;IS R4 CORRECT
6605 026760  001401                         BEQ     3$                      ;YES GO ON
6606 026762  104001                         ERROR   +1              ;CPU ERROR
6607                                                                 ;NO GO TO ERROR
6608 026764  020527  000000          3$:    CMP     R5,#0                   ;IS R5 CORRECT
6609 026770  001401                         BEQ     4$                      ;YES GO ON
6610 026772  104001                         ERROR   +1              ;CPU ERROR
6611                                                                 ;NO GO TO ERROR
6612 026774  012703  052525          4$:    MOV     #52525,R3               ;SETUP R3
6613 027000  005002                         CLR     R2                      ;SETUP R2
6614 027002  012704  164731                 MOV     #164731,R4              ;SETUP R4
6615 027006  000277                         SCC                             ;SET ALL CC BITS
6616 027010  073327  000023                 ASHC    #23,R3                  ; TEST INSTRUCTION
6617 027014  023727  177776  000012         CMP     @#177776,#12            ;IS PS CORRECT
6618 027022  001401                         BEQ     5$                      ;YES GO ON
6619 027024  104001                         ERROR   +1              ;CPU ERROR
6620                                                                 ;NO GO TO ERROR
6621 027026  020227  000000          5$:    CMP     R2,#0                   ;IS R2 CORRECT
6622 027032  001401                         BEQ     6$                      ;YES GO ON
6623 027034  104001                         ERROR   +1              ;CPU ERROR
6624                                                                 ;NO GO TO ERROR
6625 027036  020327  000000          6$:    CMP     R3,#0                   ;IS R3 CORRECT
6626 027042  001401                         BEQ     7$                      ;YES GO ON
6627 027044  104001                         ERROR   +1              ;CPU ERROR
6628                                                                 ;NO GO TO ERROR
6629 027046  020427  164731          7$:    CMP     R4,#164731              ;IS R4 CORRECT
6630 027052  001401                         BEQ     8$                      ;YES GO ON
6631 027054  104001                         ERROR   +1              ;CPU ERROR
6632                                                                 ;NO GO TO ERROR
6633                                                                 ;
6634                                                                 ;
6635 027056  012701  027166          8$:    MOV     #TE130A,R1              ;SETUP POINTERS TO TABLES
6636
6637 027062  010104                  9$:    MOV     R1,R4                   ;SAVE A COPY OF R1
6638 027064  016102  000002                 MOV     2(R1),R2                ;SETUP R2
```

```
********** DOUBLE OPERAND TESTS *****************

 6639 027070  016103  000004                MOV     4(R1),R3                ;SETUP R3
 6640 027074  000277                         SCC                             ;SET ALL CC BITS
 6641 027076  073211                         ASHC    (R1),R2         ; TEST INSTRUCTION
 6642 027100  023761  177776  000006         CMP     @#177776,6(R1)          ;IS PS CORRECT
 6643 027106  001401                         BEQ     10$                     ;YES GO ON
 6644 027110  104001                         ERROR   .+1             ;CPU ERROR
 6645                                                                ;NO GO TO ERROR
 6646 027112  026102  000010          10$:   CMP     10(R1),R2               ;IS R2 CORRECT
 6647 027116  001401                         BEQ     11$                     ;YES GO ON
 6648 027120  104001                         ERROR   .+1             ;CPU ERROR
 6649                                                                ;NO GO TO ERROR
 6650 027122  026103  000012          11$:   CMP     12(R1),R3               ;IS R3 CORRECT
 6651 027126  001401                         BEQ     12$                     ;YES GO ON
 6652 027130  104001                         ERROR   .+1             ;CPU ERROR
 6653                                                                ;NO GO TO ERROR
 6654 027132  020401                  12$:   CMP     R4,R1                   ;IS R1 CORRECT
 6655 027134  001402                         BEQ     13$                     ;YES GO ON
 6656 027136  104001                         ERROR   .+1             ;CPU ERROR
 6657                                                                ;NO GO TO ERROR
 6658 027140  010401                         MOV     R4,R1
 6659 027142  021114                  13$:   CMP     (R1),(R4)               ;IS SOURCE CORRECT
 6660 027144  001401                         BEQ     14$                     ;YES GO ON
 6661 027146  104001                         ERROR   .+1             ;CPU ERROR
 6662                                                                ;NO GO TO ERROR
 6663                                                                ;POSSIBLE SOURCE CODE CORRUPTION
 6664 027150  062701  000014          14$:   ADD     #14,R1                  ;GO TO NEXT TEST
 6665 027154  020127  027576                 CMP     R1,#FIN130              ;ARE WE DONE
 6666 027160  001340                         BNE     9$                      ;NO GO TO 9$
 6667
 6668
 6669 027162  000167  000410                JMP     FIN130
 6670                                 ;
 6671                                 ;
 6672 027166  177700                  TE130A: .WORD  177700          ;SOURCE
 6673 027170  100125                          .WORD  100125          ;DESTINATION WORD 1
 6674 027172  177777                          .WORD  177777          ;DESTINATION WORD 2
 6675 027174  000010                          .WORD  10              ;TEST PSW
 6676 027176  100125                          .WORD  100125          ;RESULT WORD 1
 6677 027200  177777                          .WORD  177777          ;RESULT WORD 2
 6678
 6679 027202  177777                          .WORD  177777          ;SOURCE
 6680 027204  000001                          .WORD  1               ;DESTINATION WORD 1
 6681 027206  000000                          .WORD  0               ;DESTINATION WORD 2
 6682 027210  000000                          .WORD  0               ;TEST PSW
 6683 027212  000000                          .WORD  0               ;RESULT WORD 1
 6684 027214  100000                          .WORD  100000          ;RESULT WORD 2
 6685
 6686 027216  177701                          .WORD  177701          ;SOURCE
 6687 027220  047777                          .WORD  47777           ;DESTINATION WORD 1
 6688 027222  100000                          .WORD  100000          ;DESTINATION WORD 2
 6689 027224  000012                          .WORD  12              ;TEST PSW
 6690 027226  117777                          .WORD  117777          ;RESULT WORD 1
 6691 027230  000000                          .WORD  0               ;RESULT WORD 2
 6692
 6693 027232  177706                          .WORD  177706          ;SOURCE
 6694 027234  004256                          .WORD  4256            ;DESTINATION WORD 1
 6695 027236  177700                          .WORD  177700          ;DESTINATION WORD 2
```

********** DOUBLE OPERAND TESTS **********

```
6696 027240  000002                    .WORD   2               ;TEST PSW
6697 027242  025677                    .WORD   25677           ;RESULT WORD 1
6698 027244  170000                    .WORD   170000          ;RESULT WORD 2
6699
6700 027246  177711                    .WORD   177711          ;SOURCE
6701 027250  065700                    .WORD   65700           ;DESTINATION WORD 1
6702 027252  000012                    .WORD   12              ;DESTINATION WORD 2
6703 027254  000013                    .WORD   13              ;TEST PSW
6704 027256  100000                    .WORD   100000          ;RESULT WORD 1
6705 027260  012000                    .WORD   12000           ;RESULT WORD 2
6706
6707 027262  177737                    .WORD   177737          ;SOURCE
6708 027264  000000                    .WORD   0               ;DESTINATION WORD 1
6709 027266  000001                    .WORD   1               ;DESTINATION WORD 2
6710 027270  000004                    .WORD   4               ;TEST PSW
6711 027272  000000                    .WORD   0               ;RESULT WORD 1
6712 027274  000000                    .WORD   0               ;RESULT WORD 2
6713
6714 027276  177736                    .WORD   177736          ;SOURCE
6715 027300  000000                    .WORD   0               ;DESTINATION WORD 1
6716 027302  000001                    .WORD   1               ;DESTINATION WORD 2
6717 027304  000000                    .WORD   0               ;TEST PSW
6718 027306  040000                    .WORD   40000           ;RESULT WORD 1
6719 027310  000000                    .WORD   0               ;RESULT WORD 2
6720
6721 027312  177740                    .WORD   177740          ;SOURCE
6722 027314  100000                    .WORD   100000          ;DESTINATION WORD 1
6723 027316  000000                    .WORD   0               ;DESTINATION WORD 2
6724 027320  000011                    .WORD   11              ;TEST PSW
6725 027322  177777                    .WORD   177777          ;RESULT WORD 1
6726 027324  177777                    .WORD   177777          ;RESULT WORD 2
6727
6728 027326  177725                    .WORD   177725          ;SOURCE
6729 027330  177777                    .WORD   177777          ;DESTINATION WORD 1
6730 027332  174000                    .WORD   174000          ;DESTINATION WORD 2
6731 027334  000007                    .WORD   7               ;TEST PSW
6732 027336  000000                    .WORD   0               ;RESULT WORD 1
6733 027340  000000                    .WORD   0               ;RESULT WORD 2
6734
6735 027342  177724                    .WORD   177724          ;SOURCE
6736 027344  177777                    .WORD   177777          ;DESTINATION WORD 1
6737 027346  174000                    .WORD   174000          ;DESTINATION WORD 2
6738 027350  000011                    .WORD   11              ;TEST PSW
6739 027352  100000                    .WORD   100000          ;RESULT WORD 1
6740 027354  000000                    .WORD   0               ;RESULT WORD 2
6741
6742 027356  177733                    .WORD   177733          ;SOURCE
6743 027360  177777                    .WORD   177777          ;DESTINATION WORD 1
6744 027362  157023                    .WORD   157023          ;DESTINATION WORD 2
6745 027364  000012                    .WORD   12              ;TEST PSW
6746 027366  114000                    .WORD   114000          ;RESULT WORD 1
6747 027370  000000                    .WORD   0               ;RESULT WORD 2
6748
6749 027372  177727                    .WORD   177727          ;SOURCE
6750 027374  000000                    .WORD   0               ;DESTINATION WORD 1
6751 027376  177777                    .WORD   177777          ;DESTINATION WORD 2
6752 027400  000013                    .WORD   13              ;TEST PSW
```

********** DOUBLE OPERAND TESTS **********

```
6753 027402  177600                .WORD   177600          ;RESULT WORD 1
6754 027404  000000                .WORD   0               ;RESULT WORD 2
6755
6756 027406  177717                .WORD   177717          ;SOURCE
6757 027410  177777                .WORD   177777          ;DESTINATION WORD 1
6758 027412  000001                .WORD   1               ;DESTINATION WORD 2
6759 027414  000011                .WORD   11              ;TEST PSW
6760 027416  100000                .WORD   100000          ;RESULT WORD 1
6761 027420  100000                .WORD   100000          ;RESULT WORD 2
6762
6763 027422  177741                .WORD   177741          ;SOURCE
6764 027424  100000                .WORD   100000          ;DESTINATION WORD 1
6765 027426  000000                .WORD   0               ;DESTINATION WORD 2
6766 027430  000010                .WORD   10              ;TEST PSW
6767 027432  177777                .WORD   177777          ;RESULT WORD 1
6768 027434  177777                .WORD   177777          ;RESULT WORD 2
6769
6770 027436  177742                .WORD   177742          ;SOURCE
6771 027440  037777                .WORD   37777           ;DESTINATION WORD 1
6772 027442  177777                .WORD   177777          ;DESTINATION WORD 2
6773 027444  000005                .WORD   5               ;TEST PSW
6774 027446  000000                .WORD   0               ;RESULT WORD 1
6775 027450  000000                .WORD   0               ;RESULT WORD 2
6776
6777 027452  177742                .WORD   177742          ;SOURCE
6778 027454  077777                .WORD   77777           ;DESTINATION WORD 1
6779 027456  177777                .WORD   177777          ;DESTINATION WORD 2
6780 027460  000001                .WORD   1               ;TEST PSW
6781 027462  000000                .WORD   0               ;RESULT WORD 1
6782 027464  000001                .WORD   1               ;RESULT WORD 2
6783
6784 027466  177711                .WORD   177711          ;SOURCE
6785 027470  065600                .WORD   65600           ;DESTINATION WORD 1
6786 027472  000012                .WORD   12              ;DESTINATION WORD 2
6787 027474  000003                .WORD   3               ;TEST PSW
6788 027476  000000                .WORD   0               ;RESULT WORD 1
6789 027500  012000                .WORD   12000           ;RESULT WORD 2
6790
6791 027502  177740                .WORD   177740          ;SOURCE
6792 027504  077777                .WORD   77777           ;DESTINATION WORD 1
6793 027506  177777                .WORD   177777          ;DESTINATION WORD 2
6794 027510  000004                .WORD   4               ;TEST PSW
6795 027512  000000                .WORD   0               ;RESULT WORD 1
6796 027514  000000                .WORD   0               ;RESULT WORD 2
6797
6798 027516  177737                .WORD   177737          ;SOURCE
6799 027520  177777                .WORD   177777          ;DESTINATION WORD 1
6800 027522  177774                .WORD   177774          ;DESTINATION WORD 2
6801 027524  000011                .WORD   11              ;TEST PSW
6802 027526  177777                .WORD   177777          ;RESULT WORD 1
6803 027530  177777                .WORD   177777          ;RESULT WORD 2
6804
6805 027532  177747                .WORD   177747          ;SOURCE
6806 027534  100000                .WORD   100000          ;DESTINATION WORD 1
6807 027536  174000                .WORD   174000          ;DESTINATION WORD 2
6808 027540  000010                .WORD   10              ;TEST PSW
6809 027542  177777                .WORD   177777          ;RESULT WORD 1
```

```
********** DOUBLE OPERAND TESTS *****************

6810 027544  177700                          .WORD   177700          ;RESULT WORD 2
6811
6812 027546  177753                          .WORD   177753          ;SOURCE
6813 027550  006324                          .WORD   6324            ;DESTINATION WORD 1
6814 027552  071002                          .WORD   71002           ;DESTINATION WORD 2
6815 027554  000001                          .WORD   1               ;TEST PSW
6816 027556  000000                          .WORD   0               ;RESULT WORD 1
6817 027560  000146                          .WORD   146             ;RESULT WORD 2
6818
6819 027562  177765                          .WORD   177765          ;SOURCE
6820 027564  102351                          .WORD   102351          ;DESTINATION WORD 1
6821 027566  177231                          .WORD   177231          ;DESTINATION WORD 2
6822 027570  000011                          .WORD   11              ;TEST PSW
6823 027572  177760                          .WORD   177760          ;RESULT WORD 1
6824 027574  116477                          .WORD   116477          ;RESULT WORD 2
6825 027576                          FIN130:
6828 027576                          MSPAU:
6829
6830                                 ;       TEST THAT AUTO DEC/INC OPERATIONS USING SP ARE ON WORD BOUNDRYS
6831 027576  005006                          CLR     R6                      ;CLEAR SP
6832 027600  112667  153314                  MOVB    (R6)+,COUNT             ;TRY AUTOINC ON R6
6833 027604  022706  000002                  CMP     #2,R6                   ;VERIFY AUTO INC BY 2
6834 027610  001401                          BEQ     SPAU1                   ;BRANCH IF GOOD
6835                                                                         ;BAD AUTO-INC
6836 027612  104001                          ERROR   +1              ;CPU ERROR
6837 027614  005006                  SPAU1:  CLR     R6                      ;CLEAR R6
6838 027616  112667  153276                  MOVB    (R6)+,COUNT
6839 027622  112667  153272                  MOVB    (R6)+,COUNT             ;DOUBLE BYTE AUTO-INC
6840 027626  022706  000004                  CMP     #4,R6           ;VERIFY RESULT
6841 027632  001401                          BEQ     SPAU2           ;BRANCH IF GOOD
6842 027634  104001                          ERROR   +1              ;CPU ERROR
6843                                                         ;BAD DOUBLE AUTO-INC
6844 027636  012706  001000          SPAU2:  MOV     #STBOT,R6               ;LOAD R6
6845 027642  114667  153252                  MOVB    -(R6),COUNT             ; TEST AUTO-DEC
6846 027646  022706  000776                  CMP     #776,R6         ;VERIFY RESULT
6847 027652  001401                          BEQ     SPAU3           ;BRANCH IF GOOD
6848 027654  104001                          ERROR   +1              ;CPU ERROR
6849
6850 027656  012706  001000          SPAU3:  MOV     #STBOT,R6               ;LOAD R6
6851 027662  114667  153232                  MOVB    -(R6),COUNT             ; TEST AUTO-DEC
6852 027666  114667  153226                  MOVB    -(R6),COUNT             ; TEST AUTO-DEC
6853 027672  022706  000774                  CMP     #774,R6         ;VERIFY RESULT
6854 027676  001401                          BEQ     SPAU4           ;BRANCH IF GOOD
6855 027700  104001                          ERROR   +1              ;CPU ERROR
6856
6857 027702  005006                  SPAU4:  CLR     R6              ; TEST AUTO-INC ON SOP
6858 027704  105726                          TSTB    (R6)+           ; TEST AUTO-INC
6859 027706  020627  000002                  CMP     R6,#2
6860 027712  001401                          BEQ     SPAU5           ;BRANCH IF GOOD
6861 027714  104001                          ERROR   +1              ;CPU ERROR
6862
6863 027716  012706  001000          SPAU5:  MOV     #STBOT,R6       ;LOAD R6
6864 027722  105746                          TSTB    -(R6)           ; TEST AUTO-DEC
6865 027724  022706  000776                  CMP     #776,R6         ;VERIFY RESULT
6866 027730  001401                          BEQ     SPAU6           ;BRANCH IF GOOD
6867 027732  104001                          ERROR   +1              ;CPU ERROR
6868
```

```
********** DOUBLE OPERAND TESTS *****************

 6869 027734  012706  001000          SPAU6:  MOV     #STBOT,R6
 6870
 6872                                 ;
 6874 027740                          MTRY:
 6875
 6876                                 ;       VERIFY YELLOW ZONE TRAP ON AUTO DEC OF R6
 6877 027740  005067  150022                  CLR     CPEREG          ;INIT CPU ERROR REGISTER
 6878 027744  012706  000150                  MOV     #150,R6         ;LOAD R6 WITH A VALUE THAT WILL
 6879                                                                 ;CAUSE A YELLOW STACK TRAP(IE. <400)
 6880 027750  016767  150030  153034          MOV     4,SLOC00        ;SAVE VECTOR
 6881 027756  012767  030014  150020          MOV     #MTRYA,4        ;SETUP THE STACK OVERFLOW TRAP POINTER
 6882 027764  016701  150156                  MOV     146,R1          ;SAVE VECTOR
 6883 027770  016702  150150                  MOV     144,R2          ;SAVE VECTOR
 6884 027774  016703  150142                  MOV     142,R3          ;SAVE VECTOR
 6885 030000  005067  150142                  CLR     146             ;JUST AS A PRECAUTION
 6886 030004  005046                          CLR     -(R6)           ;CAUSE A STACK OVERFLOW TRAP
 6887 030006  012706  001000                  MOV     #STBOT,R6       ;RESTORE R6 FOR ERROR CALL
 6888 030012  104001                          ERROR   +1              ;CPU ERROR
 6889                                                         ;OVERFLOW TRAP FAILED
 6890 030014                          MTRYA:
 6891 030014  022767  000010  147744          CMP     #BIT03,CPEREG   ;WAS CPU ERROR REG SET PROPERLY?
 6892 030022  001003                          BNE     1$              ;GO TO ERROR IF NOT
 6893 030024  020627  000142                  CMP     R6,#142         ;VERIFY CORRECT DECRIMENT OF R6
 6894 030030  001401                          BEQ     MTRYB           ;BRANCH IF GOOD
 6895 030032  104001                  1$:     ERROR   +1              ;CPU ERROR
 6896                                                         ;R6 IMPROPERLY DECRIMENTED
 6897                                                         ;OR CPU ERROR REGISTER NOT CORRECT
 6898 030034                          MTRYB:
 6899 030034  005067  147726                  CLR     CPEREG          ;CLEAR THE CPU ERROR REGISTER
 6900 030040  016767  152746  147736          MOV     SLOC00,4        ;RESTORE VECTOR
 6901 030046  010167  150074                  MOV     R1,146          ;RESTORE VECTORS
 6902 030052  010267  150066                  MOV     R2,144          ;       "
 6903 030056  010367  150060                  MOV     R3,142          ;       "
 6904 030062  012706  001000                  MOV     #STBOT,R6
 6905
 6906
 6908                                 ;
 6910 030066                          MTRYM:
 6911
 6912                                 ;       TEST STACK OVERFLOW TRAPS IN VARIOUS MODES
 6913 030066  005067  147674                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 6914 030072  012706  000400                  MOV     #400,R6                 ;SETUP OVERFLOW R6 DATA
 6915 030076  016767  147702  152706          MOV     4,SLOC00                ;SAVE VECTOR
 6916 030104  012767  030126  147672          MOV     #TRYMA,4
 6917 030112  005067  150260                  CLR     376                     ;JUST AS A PRECAUTION
 6918 030116  005046                          CLR     -(R6)                   ;CAUSE OVERFLOW TRAP
 6919 030120  012706  001000                  MOV     #STBOT,R6               ;RESTORE R6 FOR ERROR CALL
 6920 030124  104001                          ERROR   +1              ;CPU ERROR
 6921                                                                 ;NO OVERFLOW TRAP
 6922 030126                          TRYMA:
 6923 030126  005067  147634                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 6924 030132  012705  001000                  MOV     #1000,R5                ;SETUP R5 DATA
 6925 030136  012706  000400                  MOV     #400,R6                 ;SETUP OVERFLOW R6 DATA
 6926 030142  012767  030160  147634          MOV     #TRYMB,4
 6927 030150  064645                          ADD     -(R6),-(R5)             ;CAUSE OVERFLOW TRAP
 6928 030152  012706  001000                  MOV     #STBOT,R6               ;RESTORE R6 FOR ERROR CALL
 6929 030156  104001                          ERROR   +1              ;CPU ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

 6930
 6931 030160                                TRYMB:                                  ;NO OVERFLOW TRAP
 6932 030160  005067  147602                       CLR     CPEREG                          ;CLEAR CPU ERROR REGISTER
 6933 030164  012706  000150                       MOV     #150,R6                         ;SETUP OVERFLOW R6 DATA
 6934 030170  012767  030206  147606               MOV     #TRYMC,4
 6935 030176  044546                               BIC     -(R5),-(R6)                     ;CAUSE OVERFLOW TRAP
 6936 030200  012706  001000                       MOV     #STBOT,R6                       ;RESTORE R6 FOR ERROR CALL
 6937 030204  104001                               ERROR   +1                      ;CPU ERROR
 6938                                                                               ;NO OVERFLOW TRAP
 6939 030206  005067  147554                TRYMC: CLR     CPEREG                          ;CLEAR CPU ERROR REGISTER
 6940 030212  016767  152574  147564               MOV     SLOC00,4                        ;RESTORE VECTOR
 6941 030220  012706  001000                       MOV     #STBOT,R6
 6942
 6943
 6945                                       ;
 6947 030224                                MILLO:
 6948
 6949                                       ;       TEST STACK OVERFLOW ON ILLEGAL INST TRAP
 6950 030224  005067  147536                       CLR     CPEREG                          ;CLEAR CPU ERROR REGISTER
 6951 030230  012706  000400                       MOV     #400,R6                         ;SETUP FOR OVERFLOW TRAP
 6952 030234  016767  147550  152550               MOV     10,SLOC00                       ;SAVE VECTOR
 6953 030242  012767  030270  147540               MOV     #MILLOA,10                      ;SETUP ILLEGAL TRAP VECTOR
 6954 030250  016767  147530  152536               MOV     4,SLOC01                        ;SAVE VECTOR
 6955 030256  012767  030276  147520               MOV     #MILLOB,4                       ;SETUP OVERFLOW TRAP VECTOR
 6956 030264  000077                               77                                      ;UNUSED INSTRUCTION TRAP
 6957 030266  000240                               NOP
 6958 030270  012706  001000                MILLOA: MOV     #STBOT,R6                       ;RESTORE R6 FOR ERROR CALL
 6959 030274  104001                               ERROR   +1                      ;CPU ERROR
 6960                                                                               ;UNUSED INSTRUCTION TRAP
 6961 030276                                MILLOB:
 6962 030276  016767  152512  147500               MOV     SLOC01,4                        ;RESTORE VECTOR
 6963 030304  016767  152502  147476               MOV     SLOC00,10                       ;RESTORE VECTOR
 6964 030312  005067  147450                       CLR     CPEREG                          ;CLEAR CPU ERROR REGISTER
 6965 030316  012706  001000                       MOV     #STBOT,R6                       ;RESTORE R6
 6966
 6967
 6968
 6970                                       ;
 6972 030322                                MIOTO:
 6973
 6974                                       ;       TEST STACK OVERFLOW ON IOT TRAP
 6975 030322  005067  147440                       CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 6976 030326  012706  000400                       MOV     #400,R6                         ;SETUP STACK FOR OVERFLOW
 6977 030332  016767  147462  152452               MOV     20,SLOC00                       ;SAVE OLD IOT VECTOR
 6978 030340  012767  030366  147452               MOV     #IOTOA,20                       ;SETUP ERROR ACTION ON IOT
 6979 030346  016767  147432  152440               MOV     4,SLOC01                        ;SAVE VECTOR
 6980 030354  012767  030374  147422               MOV     #IOTOB,4                        ;SETUP CORRECT TRAP VECTOR FOR
 6981                                                                                       ;OVERFLOW
 6982 030362  000004                               IOT                                     ; TEST INSTRUCTION
 6983 030364  000240                               NOP
 6984 030366  012706  001000                IOTOA: MOV     #STBOT,R6                       ;RESTORE R6 FOR ERROR CALL
 6985 030372  104001                               ERROR   +1                      ;CPU ERROR
 6986                                                                               ;FAILURE OF STACK OVERFLOW
 6987 030374                                IOTOB:
 6988 030374  005067  147366                       CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 6989 030400  012706  001000                       MOV     #STBOT,R6
 6990 030404  016767  152404  147372               MOV     SLOC01,4                        ;RESTORE VECTOR
```

```
********** DOUBLE OPERAND TESTS *****************

 6991 030412  016767  152374  147400          MOV     SLOC00,20                       ;RESTORE TRAP VECTOR
 6992
 6994                                  ;
 6995                                  ;
 6997 030420                           MEMT0:
 6998
 6999                                  ;       TEST STACK OVERFLOW ON EMT TRAP
 7000 030420  005067  147342                    CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 7001 030424  012706  000400                    MOV     #400,R6                         ;SETUP STACK FOR OVERFLOW
 7002 030430  016767  147374  152354            MOV     30,SLOC00                       ;SAVE OLD EMT VECTOR
 7003 030436  012767  030464  147364            MOV     #EMT0A,30                       ;SETUP ERROR ACTION ON EMT
 7004 030444  016767  147334  152342            MOV     4,SLOC01                        ;SAVE VECTOR
 7005 030452  012767  030472  147324            MOV     #EMT0B,4                        ;SETUP CORRECT TRAP VECTOR FOR
 7006                                                                                   ;OVERFLOW
 7007 030460  104000                            EMT                                     ; TEST INSTRUCTION
 7008 030462  000240                            NOP
 7009 030464  012706  001000          EMT0A:    MOV     #STBOT,R6                       ;RESTORE R6 FOR ERROR CALL
 7010 030470  104001                            ERROR   +1              ;CPU ERROR
 7011                                                                           ;FAILURE OF STACK OVERFLOW
 7012 030472                           EMT0B:
 7013 030472  016767  152314  147330            MOV     SLOC00,30                       ;RESTORE TRAP VECTOR
 7014 030500  016767  152310  147276            MOV     SLOC01,4                        ;RESTORE VECTOR
 7015 030506  005067  147254                    CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 7016 030512  012706  001000                    MOV     #STBOT,R6
 7017
 7020 030516                           MTRP0:
 7021
 7022                                  ;       TEST STACK OVERFLOW ON TRAP
 7023 030516  005067  147244                    CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 7024 030522  012706  000400                    MOV     #400,R6                         ;SETUP STACK FOR OVERFLOW
 7025 030526  016767  147302  152256            MOV     34,SLOC00                       ;SAVE OLD TRP VECTOR
 7026 030534  012767  030562  147272            MOV     #TRP0A,34                       ;SETUP ERROR ACTION ON TRP
 7027 030542  016767  147236  152244            MOV     4,SLOC01                        ;SAVE VECTOR
 7028 030550  012767  030570  147226            MOV     #TRP0B,4                        ;SETUP CORRECT TRAP VECTOR FOR
 7029                                                                                   ;OVERFLOW
 7030 030556  104400                            TRAP                                    ; TEST INSTRUCTION
 7031 030560  000240                            NOP
 7032 030562  012706  001000          TRP0A:    MOV     #STBOT,R6                       ;RESTORE R6 FOR ERROR CALL
 7033 030566  104001                            ERROR   +1              ;CPU ERROR
 7034                                                                           ;FAILURE OF STACK OVERFLOW
 7035 030570                           TRP0B:
 7036 030570  016767  152216  147236            MOV     SLOC00,34                       ;RESTORE TRAP VECTOR
 7037 030576  016767  152212  147200            MOV     SLOC01,4                        ;RESTORE VECTOR
 7038 030604  005067  147156                    CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 7039 030610  012706  001000                    MOV     #STBOT,R6
 7040
 7042                                  ;
 7043                                  ;
 7045 030614                           MBPT0:
 7046
 7047                                  ;       TEST STACK OVERFLOW ON BPT
 7048 030614  005067  147146                    CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 7049 030620  012706  000400                    MOV     #400,R6                         ;SETUP STACK FOR OVERFLOW
 7050 030624  016767  147164  152160            MOV     14,SLOC00                       ;SAVE OLD BPT VECTOR
 7051 030632  012767  030660  147154            MOV     #BPT0A,14                       ;SETUP ERROR ACTION ON BPT
 7052 030640  016767  147140  152146            MOV     4,SLOC01                        ;SAVE VECTOR
 7053 030646  012767  030666  147130            MOV     #BPT0B,4                        ;SETUP CORRECT TRAP VECTOR FOR
```

```
********** DOUBLE OPERAND TESTS *****************

 7054
 7055 030654  000003                        BPT                                ;OVERFLOW
 7056 030656  000240                        NOP                                ; TEST INSTRUCTION
 7057 030660  012706  001000        BPTOA:  MOV     #STBOT,R6                  ;RESTORE R6 FOR ERROR CALL
 7058 030664  104001                        ERROR   +1              ;CPU ERROR
 7059                                                               ;FAILURE OF STACK OVERFLOW
 7060 030666                        BPTOB:
 7061 030666  005067  147074                CLR     CPEREG                     ;CLEAR CPU ERROR REGISTER
 7062 030672  016767  152114  147114        MOV     SLOC00,14                  ;RESTORE TRAP VECTOR
 7063 030700  016767  152110  147076        MOV     SLOC01,4                   ;RESTORE VECTOR
 7064 030706  012706  001000                MOV     #STBOT,R6
 7065
 7067                               ;
 7068                               ;
 7070 030712                        MILAO:
 7071
 7072                               ;       TEST STACK OVERFLOW AND ILLEGAL JMP INSTRUCTION
 7073 030712  005067  147050                CLR     CPEREG                     ;CLEAR CPU ERROR REGISTER
 7074 030716  012706  000400                MOV     #400,R6                    ;SETUP STACK FOR OVERFLOW
 7075 030722  016767  147062  152062        MOV     10,SLOC00                  ;SAVE OLD ILLEGAL INST. VECTOR
 7076 030730  012767  030760  147052        MOV     #ILAOA,10                  ;SETUP ERROR ACTION ILLEGAL OPCODE
 7077 030736  016767  147042  152050        MOV     4,SLOC01                   ;SAVE VECTOR
 7078 030744  012767  030766  147032        MOV     #ILBOB,4                   ;SETUP CORRECT TRAP VECTOR FOR
 7079                                                                          ;OVERFLOW
 7080 030752  005001                        CLR     R1
 7081 030754  000101                        JMP     R1                         ; TEST INSTRUCTION
 7082 030756  000240                        NOP
 7083 030760  012706  001000        ILAOA:  MOV     #STBOT,R6                  ;RESTORE R6 FOR ERROR CALL
 7084 030764  104001                        ERROR   +1              ;CPU ERROR
 7085                                                               ;FAILURE OF STACK OVERFLOW
 7086 030766                        ILBOB:
 7087 030766  016767  152022  147010        MOV     SLOC01,4                   ;RESTORE VECTOR
 7088 030774  016767  152012  147006        MOV     SLOC00,10                  ;RESTORE TRAP VECTOR
 7089 031002  005067  146760                CLR     CPEREG                     ;CLEAR CPU ERROR REGISTER
 7090 031006  012706  001000                MOV     #STBOT,R6
 7091
 7093                               ;
 7094                               ;
 7096 031012                        MILLBO:
 7097
 7098                               ;       TEST STACK OVERFLOW ON ILLEGAL JSR INST
 7099 031012  012706  000400                MOV     #400,R6                    ;SETUP STACK FOR OVERFLOW
 7100 031016  016767  146766  151766        MOV     10,SLOC00                  ;SAVE OLD VECTOR
 7101 031024  012767  031054  146756        MOV     #ILLBOA,10                 ;SETUP ERROR ACTION ON ILL. OPCODE
 7102 031032  016767  146746  151754        MOV     4,SLOC01                   ;SAVE VECTOR
 7103 031040  012767  031062  146736        MOV     #ILLBOB,4                  ;SETUP CORRECT TRAP VECTOR FOR
 7104                                                                          ;OVERFLOW
 7105 031046  005001                        CLR     R1
 7106 031050  004501                        JSR     R5,R1                      ; TEST INSTRUCTION
 7107 031052  000240                        NOP
 7108 031054  012706  001000        ILLBOA: MOV     #STBOT,R6                  ;RESTORE R6 FOR ERROR CALL
 7109 031060  104001                        ERROR   +1              ;CPU ERROR
 7110                                                               ;FAILURE OF STACK OVERFLOW
 7111 031062                        ILLBOB:
 7112 031062  016767  151726  146714        MOV     SLOC01,4                   ;RESTORE VECTOR
 7113 031070  016767  151716  146712        MOV     SLOC00,10                  ;RESTORE TRAP VECTOR
 7114 031076  012706  001000                MOV     #STBOT,R6
```

```
********** DOUBLE OPERAND TESTS *****************

                                        ;
                                        ;
031102                                  MSTO:
                                        ;       TEST FOR FALSE STACK OVERFLOW
031102  016767  146676  151702                MOV     4,SLOC00                ;SAVE VECTOR
031110  012767  031156  146666                MOV     #MSTOE,4                ;ANTICIPATE OVERFLOW ERROR
031116  012706  001002                        MOV     #1002,R6                ;SETUP LEGAL R6
031122  005746                                TST     -(R6)                   ;TRY TO CAUSE STACK OVERFLOW
031124  012706  002002                        MOV     #2002,R6                ;SETUP LEGAL R6
031130  005746                                TST     -(R6)                   ;TRY TO CAUSE STACK OVERFLOW
031132  012706  004002                        MOV     #4002,R6                ;SETUP LEGAL R6
031136  005746                                TST     -(R6)                   ;TRY TO CAUSE STACK OVERFLOW
031140  012706  010002                        MOV     #10002,R6               ;SETUP LEGAL R6
031144  005746                                TST     -(R6)                   ;TRY TO CAUSE STACK OVERFLOW
031146  012706  100402                        MOV     #100402,R6              ;SETUP  LEGAL R6
031152  005746                                TST     -(R6)                   ;TRY TO CAUSE STACK OVERFLOW
031154  000403                                BR      MSTOEE                  ;EXIT MODULE
031156  012706  001000                MSTOE:  MOV     #STBOT,R6               ;RESTORE R6 FOR ERROR CALL
031162  104001                                ERROR   +1              ;CPU ERROR
                                                                        ;STACK OVERFLOW ERROR
031164  016767  151622  146612  MSTOEE: MOV     SLOC00,4                        ;RESTORE VECTOR
031172  012706  001000                        MOV     #STBOT,R6

                                        ;
                                        ;
031176                                  MTT:

                                        ;       TEST T-BIT TRAPS
031176  012706  001000                        MOV     #STBOT,R6               ;SETUP STACK
031202  016767  146606  151602                MOV     14,SLOC00               ;SAVE OLD T-BIT VECTOR
031210  012746  000020                        MOV     #20,-(R6)               ;PUSH T-BIT
031214  012746  031232                        MOV     #MTTA,-(R6)             ;SETUP ERROR TRAP VECTOR
031220  012767  031234  146566                MOV     #MTTB,14                ;SETUP NEW T-BIT VECTOR
031226  000002                                RTI                             ;CAUSE A T BIT SET IN PSW
031230  104001                                ERROR   +1              ;CPU ERROR
                                                                        ;SHOULD NEVER BE EXECUTED
031232  104001                        MTTA:   ERROR   +1              ;CPU ERROR
                                                                        ;DIDNT TAKE CORRECT TRAP
031234  022706  000774                MTTB:   CMP     #STBOT-4,R6             ;VERIFY SP DECIRMENT
031240  001401                                BEQ     MTTD                    ;BRANCH IF GOOD
031242  104001                                ERROR   +1              ;CPU ERROR
                                                                        ;BAD SP
031244  021627  031232                MTTD:   CMP     (R6),#MTTA              ;VERIFY PC SAVED ON STACK
031250  001401                                BEQ     MTTE                    ;BRANCH IF GOOD
031252  104001                                ERROR   +1              ;CPU ERROR
                                                                        ;INCORRECT PC ON STACK
031254                                  MTTE:
031254  016767  151532  146532                MOV     SLOC00,14               ;RESTORE VECTOR 14

031262  012706  001000                        MOV     #STBOT,R6

                                        ;
031266                                  MTTS:

                                        ;       TEST T-BIT TRAPS WITH RTT
```

```
********** DOUBLE OPERAND TESTS *****************

 7178 031266  012706  001000                    MOV     #STBOT,R6               ;SETUP STACK
 7179 031272  016767  146516  151512            MOV     14,SLOC00               ;SAVE OLD T-BIT VECTOR
 7180 031300  012746  000020                    MOV     #20,-(R6)               ;PUSH T-BIT
 7181 031304  012746  031322                    MOV     #MTTSA,-(R6)            ;SETUP ERROR TRAP VECTOR
 7182 031310  012767  031326  146476            MOV     #MTTSB,14               ;SETUP NEW T-BIT VECTOR
 7183 031316  000006                            RTT                             ;CAUSE A T BIT SET IN PSW
 7184 031320  104001                            ERROR   +1              ;CPU ERROR
 7185                                                                   ;SHOULD NEVER BE EXECUTED
 7186 031322  000240                    MTTSA:  NOP                             ;RTT WILL EXECUTE THIS INSTRUCTION
 7187                                                                           ;WITH A T-BIT TRAP
 7188 031324  104001                    MTTSQ:  ERROR   +1              ;CPU ERROR
 7189                                                                   ;DIDNT TAKE CORRECT TRAP
 7190 031326  022706  000774            MTTSB:  CMP     #STBOT-4,R6             ;VERIFY SP DECIRMENT
 7191 031332  001401                            BEQ     MTTSD                   ;BRANCH IF GOOD
 7192 031334  104001                            ERROR   +1              ;CPU ERROR
 7193                                                                   ;BAD SP
 7194 031336  021627  031324            MTTSD:  CMP     (R6),#MTTSQ             ;VERIFY PC SAVED ON STACK
 7195 031342  001401                            BEQ     MTTSE                   ;BRANCH IF GOOD
 7196 031344  104001                            ERROR   +1              ;CPU ERROR
 7197                                                                   ;INCORRECT PC ON STACK
 7198 031346                            MTTSE:
 7199 031346  016767  151440  146440            MOV     SLOC00,14               ;RESTORE VECTOR 14
 7200 031354  012706  001000                    MOV     #STBOT,R6
 7201
 7204 031360                            MTTR.
 7205
 7206                                   ;       TEST OLD STATUS ON T-BIT TRAP
 7207 031360  012706  001000                    MOV     #STBOT,R6               ;SETUP STACK
 7208 031364  016767  146424  151420            MOV     14,SLOC00               ;SAVE OLD  VECTOR
 7209 031372  012746  000020                    MOV     #20,-(R6)               ;PUSH T-BIT
 7210 031376  012746  031424                    MOV     #MTTRA,-(R6)            ;SETUP ERROR TRAP VECTOR
 7211 031402  012767  031426  146404            MOV     #MTTRB,14               ;SETUP NEW T-BIT VECTOR
 7212 031410  012767  000357  146360            MOV     #357,PS                 ;SET PRIORITY AND COND C
 7213 031416  000277                            SCC
 7214 031420  000002                            RTI
 7215 031422  104001                            ERROR   +1              ;CPU ERROR
 7216                                                                   ;SHOULD NEVER EXECUTE
 7217 031424  104001                    MTTRA:  ERROR   +1              ;CPU ERROR
 7218                                                                   ;DIDNT TAKE CORRECT TRAP
 7219 031426  026727  147344  000020    MTTRB:  CMP     STBOT-2,#20             ;VERIFY PSW ON STACK
 7220 031434  001401                            BEQ     MTTRC                   ;BRANCH IF CORRECT STATUS
 7221 031436  104001                            ERROR   +1              ;CPU ERROR
 7222                                                                   ;BAD STATUS ON STACK
 7223 031440  012706  001000            MTTRC:  MOV     #STBOT,R6               ;SETUP STACK
 7224 031444  012746  000377                    MOV     #377,-(R6)              ;PUSH T-BIT
 7225 031450  012746  031476                    MOV     #MTTRD,-(R6)            ;SETUP ERROR TRAP VECTOR
 7226 031454  012767  031500  146332            MOV     #MTTRE,14               ;SETUP NEW T-BIT VECTOR
 7227 031462  012767  000000  146306            MOV     #0,PS                   ;CLEAR PRIORITY
 7228 031470  000257                            CCC                             ;CLEAR CONDITION CODES
 7229 031472  000002                            RTI
 7230 031474  104001                            ERROR   +1              ;CPU ERROR
 7231                                                                   ;SHOULD NEVER EXECUTE
 7232 031476  104001                    MTTRD:  ERROR   +1              ;CPU ERROR
 7233                                                                   ;DIDNT TAKE CORRECT TRAP
 7234 031500  026727  147272  000377    MTTRE:  CMP     STBOT-2,#377            ;VERIFY OLD PSW ON STACK
 7235 031506  001401                            BEQ     MTTRF                   ;BRANCH IF GOOD
 7236 031510  104001                            ERROR   +1              ;CPU ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

                                                                ;OLD PSW INCORRECT
                                MTTRF:
 031512  016767  151274  146274     MOV     SLOC00,14               ;RESTORE VECTOR
 031520  012706  001000             MOV     #STBOT,R6
                                ;
 031524                          MRT:
                                ;       TEST RESERVED INST TRAP
 031524  012706  001000             MOV     #STBOT,R6               ;SETUP STACK
 031530  016767  146254  151254     MOV     10,SLOC00               ;SAVE OLD  VECTOR
 031536  012767  031550  146244     MOV     #MRTB,10                ;SETUP NEW RESERVED VECTOR
 031544  000077                     77
 031546  104001                 MRTA:  ERROR   +1                  ;CPU ERROR
                                                                ;DIDNT TAKE CORRECT TRAP
 031550  022706  000774         MRTB:  CMP     #STBOT-4,R6             ;VERIFY SP DECRIMENT
 031554  001401                     BEQ     MRTE                    ;BRANCH IF GOOD
 031556  104001                     ERROR   +1                 ;CPU ERROR
                                                                ;BAD PC ON STACK
 031560  021627  031546         MRTE:  CMP     (R6),#MRTA              ;VERFY PROPER PC ON STACK
 031564  001401                     BEQ     MRTF                    ;BRANCH IF GOOD
 031566  104001                     ERROR   +1                 ;CPU ERROR
                                                                ;INCORRECT PC ON STACK
 031570                          MRTF:
 031570  016767  151216  146212     MOV     SLOC00,10               ;RESTORE TRAP VECTOR
 031576  012706  001000             MOV     #STBOT,R6
                                ;
                                ;
 031602                          MRTO:
                                ;       TEST OLD STATUS ON RESERVED INST TRAP
 031602  012706  001000             MOV     #STBOT,R6               ;SETUP STACK
 031606  016767  146176  151176     MOV     10,SLOC00               ;SAVE OLD  VECTOR
 031614  012767  031634  146166     MOV     #MRTOB,10               ;SETUP NEW VECTOR
 031622  005067  146150             CLR     PS                      ;CLEAR PRIORITY AND COND C
 031626  000257                     CCC
 031630  000077                     77
 031632  104001                 MRTOA: ERROR   +1                  ;CPU ERROR
                                                                ;DIDNT TAKE CORRECT TRAP
 031634  026727  147136  000000 MRTOB:  CMP     STBOT-2,#0              ;VERIFY PSW ON STACK
 031642  001401                     BEQ     MRTOC                   ;BRANCH IF CORRECT STATUS
 031644  104001                     ERROR   +1                 ;CPU ERROR
                                                                ;BAD STATUS ON STACK
 031646  012706  001000         MRTOC: MOV     #STBOT,R6               ;SETUP STACK
 031652  012767  031674  146130     MOV     #MRTOE,10               ;SET UP TRAP VECTOR
 031660  012767  000357  146110     MOV     #357,PS                 ;SET PRIORITY
 031666  000277                     SCC                             ;SET CONDITION CODES
 031670  000077                     77                              ;RESERVED INSTRUCTION

 031672  104001                 MRTOD: ERROR   +1                  ;CPU ERROR
                                                                ;DIDNT TAKE CORRECT TRAP
 031674  026727  147076  000357 MRTOE:  CMP     STBOT-2,#357            ;VERIFY OLD PSW ON STACK
 031702  001401                     BEQ     MRTOF                   ;BRANCH IF GOOD
 031704  104001                     ERROR   +1                 ;CPU ERROR
                                                                ;OLD PSW INCORRECT
 031706                          MRTOF:
```

```
********** DOUBLE OPERAND TESTS *****************

 7298 031706  016767  151100  146074          MOV     SLOC00,10               ;RESOTRE TRAP VECTOR
 7299 031714  012706  001000                  MOV     #STBOT,R6
 7300
 7303 031720                          MTP:
 7304
 7305                                 ;       TEST TRAP INST
 7306 031720  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
 7307 031724  016767  146104  151060          MOV     34,SLOC00               ;SAVE OLD  VECTOR
 7308 031732  012767  031752  146074          MOV     #MTPB,34                ;SETUP NEW TRAP VECTOR
 7309 031740  005067  146032                  CLR     PS                      ;CLEAR PRIORITY ABND COND C
 7310 031744  000257                          CCC
 7311 031746  104400                          TRAP
 7312 031750  104001                  MTPR:   ERROR   +1              ;CPU ERROR
 7313                                                                 ;DIDNT TAKE CORRECT TRAP
 7314 031752  022706  000774          MTPB:   CMP     #STBOT-4,R6             ;VERIFY SP DECRIMENT
 7315 031756  001401                          BEQ     MTPQ                    ;BRANCH IF GOOD
 7316 031760  104001                          ERROR   +1              ;CPU ERROR
 7317                                                                 ;BAD PC ON STACK
 7318 031762  021627  031750          MTPQ:   CMP     (R6),#MTPR              ;VERFY PROPER PC ON STACK
 7319 031766  001401                          BEQ     MTPF                    ;BRANCH IF GOOD
 7320 031770  104001                          ERROR   +1              ;CPU ERROR
 7321                                                                 ;INCORRECT PC ON STACK
 7322 031772                          MTPF:
 7323 031772  016767  151014  146034          MOV     SLOC00,34               ;RESTORE VECTOR
 7324 032000  012706  001000                  MOV     #STBOT,R6
 7325
 7327                                 ;
 7328                                 ;
 7330 032004                          MTPO:
 7331
 7332                                 ;       TEST OLD STATUS SAVED ON TRAP
 7333 032004  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
 7334 032010  016767  146020  150774          MOV     34,SLOC00               ;SAVE OLD  VECTOR
 7335 032016  012767  032036  146010          MOV     #MTPOB,34               ;SETUP NEW TRAP VECTOR
 7336 032024  005067  145746                  CLR     PS                      ;CLEAR PRIORITY AND COND C
 7337 032030  000257                          CCC
 7338 032032  104400                          TRAP
 7339 032034  104001                  MTPOA:  ERROR   +1              ;CPU ERROR
 7340                                                                 ;DIDNT TAKE CORRECT TRAP
 7341 032036  026727  146734  000000  MTPOB:  CMP     STBOT-2,#0              ;VERIFY PSW ON STACK
 7342 032044  001401                          BEQ     MTPOC                   ;BRANCH IF CORRECT STATUS
 7343 032046  104001                          ERROR   +1              ;CPU ERROR
 7344                                                                 ;BAD STATUS ON STACK
 7345 032050  012706  001000          MTPOC:  MOV     #STBOT,R6               ;SETUP STACK
 7346 032054  012767  032076  145752          MOV     #MTPOE,34               ;SET UP TRAP VECTOR
 7347 032062  012767  000357  145706          MOV     #357,PS                 ;SET PRIORITY
 7348 032070  000277                          SCC                             ;SET CONDITION CODES
 7349 032072  104400                          TRAP                            ;ISSUE TRAP
 7350 032074  104001                  MTPOD:  ERROR   +1              ;CPU ERROR
 7351                                                                 ;DIDNT TAKE CORRECT TRAP
 7352 032076  026727  146674  000357  MTPOE:  CMP     STBOT-2,#357            ;VERIFY OLD PSW ON STACK
 7353 032104  001401                          BEQ     MTPOF                   ;BRANCH IF GOOD
 7354 032106  104001                          ERROR   +1              ;CPU ERROR
 7355                                                                 ;OLD PSW INCORRECT
 7356 032110                          MTPOF:
 7357 032110  016767  150676  145716          MOV     SLOC00,34               ;RESTORE TRAP VECTOR
 7358 032116  012706  001000                  MOV     #STBOT,R6
```

```
********** DOUBLE OPERAND TESTS *****************

7359
7361                                    ;
7362                                    ;
7364 032122                             MTPA:
7365
7366                                    ;       TEST ALL TRAP OPCODES - SELF MODIFYING
7367 032122  005003                             CLR     R3                      ;SETUP REGISTER TO INDICATE OPCODE
7368 032124  012706  001000                     MOV     #STBOT,R6               ;SETUP STACK
7369 032130  016767  145700  150654             MOV     34,SLOC00               ;SAVE OLD  VECTOR
7370 032136  016767  145642  150650             MOV     4,SLOC01                ;SAVE IN CASE OF HALT
7371 032144  012767  032174  145632             MOV     #MTPAH,4                ;SETUP HALT TRAP
7372 032152  012767  032176  145654             MOV     #MTPAA,34               ;SETUP NEW TRAP VECTOR
7373 032160  000167  000012                     JMP     MTPAA                   ;GO INTO LOOPING CODE
7374                                    ;
7375 032164  000000                     MTPAL:  HALT                            ;SET TO A ZERO
7376 032166  104001                             ERROR   +1                      ;CPU ERROR
7377                                                                            ;TRAP INSTRUCTION FAILED TO TRAP
7378                                                                                    ;EXAMINE OPCCODE AT LOCATION MTPAL:
7379 032170  000167  000002                     JMP     MTPAA                   ;ATTEMPT TO GO ON
7380                                    ;
7381 032174  104001                     MTPAH:  ERROR   +1                      ;CPU ERROR
7382                                                                            ;ERROR, EITHER CANT MODIFY LOCATION MTPAL
7383                                                                                    ;OR TRAP INSTRUCTION FAILED
7384 032176                             MTPAA:
7385 032176  005203                             INC     R3                      ;GET NEXT OPCODE
7386
7387 032200  012706  001000                     MOV     #STBOT,R6               ;RESTORE STACK
7388 032204  020327  000400                     CMP     R3,#400                 ;SEE IF LAST OPCODE
7389 032210  001406                             BEQ     MTPAE                   ;BRANCH IF DONE
7390 032212  012767  104400  177744             MOV     #104400,MTPAL           ;TRAP OPCODE INTO LOCATION
7391 032220  060367  177740                     ADD     R3,MTPAL                ;FORM TEST OPCODE
7392 032224  000757                             BR      MTPAL                   ;EXECUTE TEST
7393 032226                             MTPAE:
7394
7395 032226  016767  150560  145600             MOV     SLOC00,34
7396 032234  016767  150554  145542             MOV     SLOC01,4                ;RESTORE VECTORS
7397
7398 032242  012706  001000                     MOV     #STBOT,R6
7399
7401                                    ;
7402                                    ;
7404 032246                             MIOT:
7405
7406                                    ;       TEST IOT TRAP
7407 032246  012706  001000                     MOV     #STBOT,R6               ;SETUP STACK
7408 032252  016767  145542  150532             MOV     20,SLOC00               ;SAVE OLD  VECTOR
7409 032260  012767  032272  145532             MOV     #MIOTB,20               ;SETUP NEW IOT VECTOR
7410 032266  000004                             IOT
7411 032270  104001                     MIOTA:  ERROR   +1                      ;CPU ERROR
7412                                                                            ;DIDNT TAKE CORRECT TRAP
7413 032272  022706  000774             MIOTB:  CMP     #STBOT-4,R6                     ;VERIFY SP DECRIMENT
7414 032276  001401                             BEQ     MIOTD                   ;BRANCH IF GOOD
7415 032300  104001                             ERROR   +1                      ;CPU ERROR
7416                                                                            ;BAD PC ON STACK
7417 032302  021627  032270             MIOTD:  CMP     (R6),#MIOTA                     ;VERFY PROPER PC ON STACK
7418 032306  001401                             BEQ     MIOTF                   ;BRANCH IF GOOD
7419 032310  104001                             ERROR   +1                      ;CPU ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

 7420                                                          ;INCORRECT PC ON STACK
 7421 032312  016767  150474  145500  MIOTF:  MOV     SLOC00,20           ;RESTORE VECTOR
 7422 032320  012706  001000          MOV     #STBOT,R6
 7423
 7426 032324                  MITO:
 7427
 7428                         ;       TEST OLD STATUS ON IOT TRAP
 7429 032324  012706  001000          MOV     #STBOT,R6           ;SETUP STACK
 7430 032330  016767  145464  150454  MOV     20,SLOC00           ;SAVE OLD  VECTOR
 7431 032336  012767  032356  145454  MOV     #MITOB,20           ;SETUP NEW IOT VECTOR
 7432 032344  005067  145426          CLR     PS                  ;CLEAR PRIORITY AND COND C
 7433 032350  000257                  CCC
 7434 032352  000004                  IOT
 7435 032354  104001          MITOA:  ERROR   +1                  ;CPU ERROR
 7436                                                          ;DIDNT TAKE CORRECT TRAP
 7437 032356  026727  146414  000000  MITOB:  CMP     STBOT-2,#0          ;VERIFY PSW ON STACK
 7438 032364  001401                  BEQ     MITOC               ;BRANCH IF CORRECT STATUS
 7439 032366  104001                  ERROR   +1                  ;CPU ERROR
 7440                                                          ;BAD STATUS ON STACK
 7441 032370  012706  001000  MITOC:  MOV     #STBOT,R6           ;SETUP STACK
 7442 032374  012767  032416  145416  MOV     #MITOE,20           ;SET UP TRAP VECTOR
 7443 032402  012767  000357  145366  MOV     #357,PS             ;SET PRIORITY
 7444 032410  000277                  SCC                         ;SET CONDITION CODES
 7445 032412  000004                  IOT
 7446 032414  104001          MITOD:  ERROR   +1                  ;CPU ERROR
 7447                                                          ;DIDNT TAKE CORRECT TRAP
 7448 032416  026727  146354  000357  MITOE:  CMP     STBOT-2,#357        ;VERIFY OLD PSW ON STACK
 7449 032424  001401                  BEQ     MITOF               ;BRANCH IF GOOD
 7450 032426  104001                  ERROR   +1                  ;CPU ERROR
 7451                                                          ;OLD PSW INCORRECT
 7452 032430                  MITOF:
 7453 032430  016767  150356  145362          MOV     SLOC00,20           ;RESTORE VECTOR
 7454 032436  012706  001000          MOV     #STBOT,R6
 7455
 7457                         ;
 7459 032442                  MET:
 7460
 7461                         ;       TEST EMULATOR TRAP INSTRUCTION  (EMT)
 7462 032442  012706  001000          MOV     #STBOT,R6           ;SETUP STACK
 7463 032446  016767  145356  150336  MOV     30,SLOC00           ;SAVE OLD  VECTOR
 7464 032454  012767  032510  145346  MOV     #METB,30            ;SETUP NEW EMT VECTOR
 7465 032462  016767  145346  150324  MOV     34,SLOC01           ;SAVE TRAP VECTOR
 7466 032470  012767  140776  145336  MOV     #$ERROR,34          ;SET UP TO HANDLE EMT ERROR
 7467 032476  104000                  EMT
 7468 032500  104400          META:   TRAP                        ;TRAP ON ERROR
 7469 032502  001057                  .WORD   559.
 7470 032504  000001                  .WORD   1                   ;CPUERR
 7471 032506  000001                  .WORD   1                   ;ERRTN
 7472                                                          ;DIDNT TAKE CORRECT TRAP
 7473 032510  022706  000774  METB:   CMP     #STBOT-4,R6         ;VERIFY SP DECRIMENT
 7474 032514  001401                  BEQ     METD                ;BRANCH IF GOOD
 7475 032516  104001                  ERROR   +1                  ;CPU ERROR
 7476                                                          ;BAD PC ON STACK
 7477 032520  021627  032500  METD:   CMP     (R6),#META          ;VERFY PROPER PC ON STACK
 7478 032524  001401                  BEQ     METF                ;BRANCH IF GOOD
 7479 032526  104001                  ERROR   +1                  ;CPU ERROR
 7480                                                          ;INCORRECT PC ON STACK
```

```
********** DOUBLE OPERAND TESTS *****************

 7481 032530  016767  150260  145276  METF:   MOV     SLOC01,34               ;RESTORE VECTOR
 7482 032536  016767  150250  145264          MOV     SLOC00,30               ;RESTORE VECTOR
 7483 032544  012706  001000                  MOV     #STBOT,R6
 7484
 7486                                 ;
 7488 032550                          METO:
 7489
 7490                                 ;       TEST OLD STATUS ON EMT TRAP
 7491 032550  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
 7492 032554  016767  145250  150230          MOV     30,SLOC00               ;SAVE OLD  VECTOR
 7493 032562  012767  032624  145240          MOV     #METOB,30               ;SETUP NEW EMT VECTOR
 7494 032570  016767  145240  150216          MOV     34,SLOC01               ;SAVE TRAP VECTOR
 7495 032576  012767  140776  145230          MOV     #$ERROR,34              ;SET UP TRAP VECTOR
 7496 032604  005067  145166                  CLR     PS                      ;CLEAR PRIORITY AND COND C
 7497 032610  000257                          CCC
 7498 032612  104000                          EMT
 7499 032614  104400                  METOA:  TRAP
 7500 032616  001062                          .WORD   562.
 7501 032620  000001                          .WORD   1                               ;CPUERR
 7502 032622  000001                          .WORD   1                               ;ERRTN
 7503                                                          ;DIDNT TAKE CORRECT TRAP
 7504 032624  026727  146146  000000  METOB:  CMP     STBOT-2,#0                      ;VERIFY PSW ON STACK
 7505 032632  001401                          BEQ     METOC                           ;BRANCH IF CORRECT STATUS
 7506 032634  104001                          ERROR   +1                      ;CPU ERROR
 7507                                                          ;BAD STATUS ON STACK
 7508 032636  012706  001000          METOC:  MOV     #STBOT,R6                       ;SETUP STACK
 7509 032642  012767  032672  145160          MOV     #METOE,30               ;SET UP TRAP VECTOR
 7510 032650  012767  000357  145120          MOV     #357,PS                 ;SET PRIORITY
 7511 032656  000277                          SCC                             ;SET CONDITION CODES
 7512 032660  104000                          EMT
 7513 032662  104400                  METOD:  TRAP
 7514 032664  001064                          .WORD   564.
 7515 032666  000001                          .WORD   1                               ;CPUERR
 7516 032670  000001                          .WORD   1                               ;ERRTN
 7517                                                          ;DIDNT TAKE CORRECT TRAP
 7518 032672  026727  146100  000357  METOE:  CMP     STBOT-2,#357                    ;VERIFY OLD PSW ON STACK
 7519 032700  001401                          BEQ     METOF                           ;BRANCH IF GOOD
 7520 032702  104001                          ERROR   +1                      ;CPU ERROR
 7521                                                          ;OLD PSW INCORRECT
 7522 032704                          METOF:
 7523 032704  016767  150104  145122          MOV     SLOC01,34               ;RESTORE VECTOR
 7524 032712  016767  150074  145110          MOV     SLOC00,30               ;RESTORE VECTOR
 7525 032720  012706  001000                  MOV     #STBOT,R6
 7526
 7528                                 ;
 7529                                 ;
 7531 032724                          MBT:
 7532
 7533                                 -;      TEST BPT TRAP
 7534 032724  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
 7535 032730  016767  145060  150054          MOV     14,SLOC00               ;SAVE OLD  VECTOR
 7536 032736  012767  032750  145050          MOV     #MBTB,14                ;SETUP NEW BPT VECTOR
 7537 032744  000003                          BPT
 7538 032746  104001                  MBTA:   ERROR   +1                      ;CPU ERROR
 7539                                                          ;DIDNT TAKE CORRECT TRAP
 7540 032750  022706  000774          MBTB:   CMP     #STBOT-4,R6                     ;VERIFY SP DECRIMENT
 7541 032754  001401                          BEQ     MBTD                            ;BRANCH IF GOOD
```

```
********** DOUBLE OPERAND TESTS *****************

 7542 032756  104001                        ERROR   +1              ;CPU ERROR
 7543                                                                ;BAD PC ON STACK
 7544 032760  021627  032746        MBTD:   CMP     (R6),#MBTA              ;VERFY PROPER PC ON STACK
 7545 032764  001401                        BEQ     MBTF                    ;BRANCH IF GOOD
 7546 032766  104001                        ERROR   +1              ;CPU ERROR
 7547                                                                ;INCORRECT PC ON STACK
 7548 032770  016767  150016  145016  MBTF:   MOV     SLOC00,14               ;RESTORE VECTOR
 7549 032776  012706  001000                MOV     #STBOT,R6
 7550
 7552                               ;
 7553                               ;
 7555 033002                        MBTO:
 7556
 7557                               ;       TEST OLD STATUS ON BPT TRAP
 7558 033002  012706  001000                MOV     #STBOT,R6               ;SETUP STACK
 7559 033006  016767  145002  147776        MOV     14,SLOC00               ;SAVE OLD  VECTOR
 7560 033014  012767  033034  144772        MOV     #MBTOB,14               ;SETUP NEW BPT VECTOR
 7561 033022  005067  144750                CLR     PS                      ;CLEAR PRIORITY AND COND C
 7562 033026  000257                        CCC
 7563 033030  000003                        BPT
 7564 033032  104001                MBTOA:  ERROR   +1              ;CPU ERROR
 7565                                                                ;DIDNT TAKE CORRECT TRAP
 7566 033034  026727  145736  000000  MBTOB:  CMP     STBOT-2,#0              ;VERIFY PSW ON STACK
 7567 033042  001401                        BEQ     MBTOC           ;BRANCH IF CORRECT STATUS
 7568 033044  104001                        ERROR   +1              ;CPU ERROR
 7569                                                                ;BAD STATUS ON STACK
 7570 033046  012706  001000        MBTOC:  MOV     #STBOT,R6               ;SETUP STACK
 7571 033052  012767  033074  144734        MOV     #MBTOE,14               ;SET UP TRAP VECTOR
 7572 033060  012767  000357  144710        MOV     #357,PS                 ;SET PRIORITY
 7573 033066  000277                        SCC                             ;SET CONDITION CODES
 7574 033070  000003                        BPT
 7575 033072  104001                MBTOD:  ERROR   +1              ;CPU ERROR
 7576                                                                ;DIDNT TAKE CORRECT TRAP
 7577 033074  026727  145676  000357  MBTOE:  CMP     STBOT-2,#357            ;VERIFY OLD PSW ON STACK
 7578 033102  001401                        BEQ     MBTOF                   ;BRANCH IF GOOD
 7579 033104  104001                        ERROR   +1              ;CPU ERROR
 7580                                                                ;OLD PSW INCORRECT
 7581 033106                        MBTOF:
 7582 033106  016767  147700  144700        MOV     SLOC00,14               ;RESTORE VECTOR
 7583 033114  012706  001000                MOV     #STBOT,R6
 7584
 7586                               ;
 7587                               ;
 7589 033120                        MIL:
 7590
 7591                               ;       TEST ILLEGAL JUMP INSTRUCTION TRAP
 7592 033120  012706  001000                MOV     #STBOT,R6               ;SETUP STACK
 7593 033124  016767  144660  147660        MOV     10,SLOC00               ;SAVE OLD  VECTOR
 7594 033132  012767  033146  144650        MOV     #MILB,10                ;SETUP NEW ILLEGAL VECTOR
 7595 033140  005001                        CLR     R1
 7596 033142  000101                        JMP     R1                      ;**TEST INSTRUCTIO
 7597 033144  104001                MILA:   ERROR   +1              ;CPU ERROR
 7598                                                                ;DIDNT TAKE CORRECT TRAP
 7599 033146  022706  000774        MILB:   CMP     #STBOT-4,R6             ;VERIFY SP DECRIMENT
 7600 033152  001401                        BEQ     MILD                    ;BRANCH IF GOOD
 7601 033154  104001                        ERROR   +1              ;CPU ERROR
 7602                                                                ;BAD PC ON STACK
```

```
********** DOUBLE OPERAND TESTS *****************

 7603 033156  021627  033144          MILD:   CMP     (R6),#MILA              ;VERFY PROPER PC ON STACK
 7604 033162  001401                          BEQ     MILF                    ;BRANCH IF GOOD
 7605 033164  104001                          ERROR   +1              ;CPU ERROR
 7606                                         ;INCORRECT PC ON STACK
 7607 033166  016767  147620  144614  MILF:   MOV     SLOC00,10               ;RESTORE VECTOR
 7608 033174  012706  001000                  MOV     #STBOT,R6
 7609
 7612 033200                          MILO:
 7613
 7614                                 ;       TEST OLD STATUS ON ILLEGAL JUMP TRAP
 7615 033200  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
 7616 033204  016767  144600  147600          MOV     10,SLOC00               ;SAVE OLD  VECTOR
 7617 033212  012767  033234  144570          MOV     #MILOB,10               ;SETUP NEW ILLEGAL VECTOR
 7618 033220  005067  144552                  CLR     PS                      ;CLEAR PRIORITY AND COND C
 7619 033224  000257                          CCC
 7620 033226  005001                          CLR     R1
 7621 033230  000101                          JMP     R1
 7622 033232  104001                  MILOA:  ERROR   +1              ;CPU ERROR
 7623                                         ;DIDNT TAKE CORRECT TRAP
 7624 033234  026727  145536  000004  MILOB:  CMP     STBOT-2,#4              ;VERIFY PSW ON STACK
 7625 033242  001401                          BEQ     MILOC                   ;BRANCH IF CORRECT STATUS
 7626 033244  104001                          ERROR   +1              ;CPU ERROR
 7627                                         ;BAD STATUS ON STACK
 7628 033246  012706  001000          MILOC:  MOV     #STBOT,R6               ;SETUP STACK
 7629 033252  012767  033274  144530          MOV     #MILOE,10               ;SET UP TRAP VECTOR
 7630 033260  012767  000357  144510          MOV     #357,PS                 ;SET PRIORITY
 7631 033266  000277                          SCC                             ;SET CONDITION CODES
 7632 033270  000101                          JMP     R1
 7633 033272  104001                  MILOD:  ERROR   +1              ;CPU ERROR
 7634                                         ;DIDNT TAKE CORRECT TRAP
 7635 033274  026727  145476  000357  MILOE:  CMP     STBOT-2,#357            ;VERIFY OLD PSW ON STACK
 7636 033302  001401                          BEQ     MILOF                   ;BRANCH IF GOOD
 7637 033304  104001                          ERROR   +1              ;CPU ERROR
 7638                                         ;OLD PSW INCORRECT
 7639 033306                          MILOF:
 7640 033306  016767  147500  144474          MOV     SLOC00,10               ;RESTORE VECTOR
 7641 033314  012706  001000                  MOV     #STBOT,R6
 7642
 7644                                 ;
 7645                                 ;
 7647 033320                          MIALL:
 7648
 7649                                 ;       TEST ILLEGAL JSR INSTRUCTION TRAP
 7650 033320  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
 7651 033324  016767  144460  147460          MOV     10,SLOC00               ;SAVE OLD  VECTOR
 7652 033332  012767  033346  144450          MOV     #MIALLB,10              ;SETUP NEW ILLEGAL VECTOR
 7653 033340  005003                          CLR     R3
 7654 033342  004303                          JSR     R3,R3
 7655 033344  104001                  MIALLA: ERROR   +1              ;CPU ERROR
 7656                                         ;DIDNT TAKE CORRECT TRAP
 7657 033346  022706  000774          MIALLB: CMP     #STBOT-4,R6             ;VERIFY SP DECRIMENT
 7658 033352  001401                          BEQ     MIALLD                  ;BRANCH IF GOOD
 7659 033354  104001                          ERROR   +1              ;CPU ERROR
 7660                                         ;BAD PC ON STACK
 7661 033356  021627  033344          MIALLD: CMP     (R6),#MIALLA            ;VERFY PROPER PC ON STACK
 7662 033362  001401                          BEQ     MIALLF                  ;BRANCH IF GOOD
 7663 033364  104001                          ERROR   +1              ;CPU ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

 7664
 7665 033366  016767  147420  144414  MIALLF: MOV     SLOC00,10               ;INCORRECT PC ON STACK
 7666 033374  012706  001000                  MOV     #STBOT,R6               ;RESTORE VECTOR
 7667
 7669                                 ;
 7670                                 ;
 7672 033400                          MJSI:
 7673
 7674                                 ;       TEST OLD STATUS ON ILLEGAL JSR TRAP
 7675 033400  012706  001000                  MOV     #STBOT,R6               ;SETUP STACK
 7676 033404  016767  144400  147400          MOV     10,SLOC00               ;SAVE OLD  VECTOR
 7677 033412  012767  033434  144370          MOV     #MJSIB,10               ;SETUP NEW VECTOR
 7678 033420  005067  144352                  CLR     PS                      ;CLEAR PRIORITY AND COND C
 7679 033424  000257                          CCC
 7680 033426  005003                          CLR     R3
 7681 033430  004303                          JSR     R3,R3
 7682 033432  104001                  MJSIA:  ERROR   +1                      ;CPU ERROR
 7683                                                                         ;DIDNT TAKE CORRECT TRAP
 7684 033434  026727  145336  000004  MJSIB:  CMP     STBOT-2,#4              ;VERIFY PSW ON STACK
 7685 033442  001401                          BEQ     MJSIC                   ;BRANCH IF CORRECT STATUS
 7686 033444  104001                          ERROR   +1                      ;CPU ERROR
 7687                                                                         ;BAD STATUS ON STACK
 7688 033446  012706  001000          MJSIC:  MOV     #STBOT,R6               ;SETUP STACK
 7689 033452  012767  033474  144330          MOV     #MJSIE,10               ;SET UP TRAP VECTOR
 7690 033460  012767  000357  144310          MOV     #357,PS                 ;SET PRIORITY
 7691 033466  000277                          SCC                             ;SET CONDITION CODES
 7692 033470  004303                          JSR     R3,R3
 7693 033472  104001                  MJSID:  ERROR   +1                      ;CPU ERROR
 7694                                                                         ;DIDNT TAKE CORRECT TRAP
 7695 033474  026727  145276  000357  MJSIE:  CMP     STBOT-2,#357            ;VERIFY OLD PSW ON STACK
 7696 033502  001401                          BEQ     MJSIF                   ;BRANCH IF GOOD
 7697 033504  104001                          ERROR   +1                      ;CPU ERROR
 7698                                                                         ;OLD PSW INCORRECT
 7699 033506                          MJSIF:
 7700 033506  016767  147300  144274          MOV     SLOC00,10               ;RESTORE VECTOR
 7701 033514  012706  001000                  MOV     #STBOT,R6
 7702
 7704                                 ;
 7705                                 ;
 7707 033520                          IOXXX:
 7708                                 ;       I/O TIME OUT TEST
 7709 033520  005067  144242                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 7710 033524  016767  144254  147260          MOV     4,SLOC00                ;SAVE VECTOR
 7711 033532  012767  033554  144244          MOV     #2$,4                   ;SET UP VECTOR TO HANDLE NXM
 7712 033540  012767  030000  144230          MOV     #30000,PS               ;INIT THE PSW TO A KNOWN STATE
 7713 033546  005737  177700                  TST     @#177700                ;TRY TO ACCESS HARDWARE ADDRESS
 7714                                                                         ;FOR GENERAL PURPOSE REG 0. THIS
 7715                                                                         ;IS NOT IMPLEMENTED ON KDJ11
 7716                                                                         ;SHOULD CAUSE TIME OUT.
 7717 033552  104001                  1$:     ERROR   +1                      ;CPU ERROR
 7718 033554  022767  000020  144204  2$:     CMP     #BIT04,CPEREG           ;IS CPU ERROR REGISTER CORRECT?
 7719 033562  001401                          BEQ     3$
 7720 033564  104001                          ERROR   +1                      ;CPU ERROR
 7721 033566  022627  033552          3$:     CMP     (SP)+,#1$               ;CHECK THAT STACK CONTAINS CORRECT ADDR.
 7722 033572  001401                          BEQ     4$                      ;
 7723 033574  104001                          ERROR   +1                      ;CPU ERROR
 7724 033576  022627  030000          4$:     CMP     (SP)+,#30000            ;IS THE PSW OK?
```

```
********** DOUBLE OPERAND TESTS *****************

 7725 033602  001401                        BEQ     5$                      ;
 7726 033604  104001                        ERROR   .1                ;CPU ERROR
 7727 033606  005067  144154          5$:   CLR     CPEREG                  ;CLEAR THE CPU ERROR REGISTER
 7728 033612  016767  147174  144164        MOV     SLOC00,4                ;RESTORE VECTOR
 7729
 7732 033620                        ODDXX:
 7741
 7742                               ;       ODD ADDRESS/ILLEGAL INST FETCH TRAP TEST
 7743                               ;;**************************************************************
                                    ;THIS PROGRAM GENERATES AN ODD ADDRESS IN THE PC. THE KDJ11 SHOULD
                                    ;TRAP THROUGH ADDR 4
 7744 033620  005067  144142                CLR     CPEREG                  ;INIT THE CPU ERROR REG
 7745 033624  016767  144154  147160        MOV     4,SLOC00                ;SAVE VECTOR
 7746 033632  012767  033666  144144        MOV     #2$,4                   ;SET UP VECTOR TO HANDLE ODD ADDR TRAP
 7747 033640  016767  144142  147146        MOV     6,SLOC01                ;SAVE VECTOR
 7748 033646  005067  144134                CLR     6                       ;INIT VECTOR
 7749 033652  012746  030000                MOV     #30000,-(SP)            ;PUSH A KNOWN PSW ON THE STACK
 7750 033656  012746  033665                MOV     #1$+1,-(SP)             ;PUSH AN ODD NUMBER ON THE STACK
 7751 033662  000002                        RTI                             ;POP ODD ADDRESS OFF STACK INTO PC
 7752                                                                       ;SHOULD TRAP HERE
 7753 033664  104001                  1$:   ERROR   .1                ;CPU ERROR
 7754 033666  022767  000100  144072  2$:   CMP     #BIT06,CPEREG           ;IS CPU ERROR REGISTER CORRECT?
 7755 033674  001401                        BEQ     3$
 7756 033676  104001                        ERROR   .1                ;CPU ERROR
 7757 033700  022726  033665          3$:   CMP     #1$+1,(SP)+             ;IS STACK CONTENTS CORRECT?
 7758 033704  001401                        BEQ     4$                      ;
 7759 033706  104001                        ERROR   .1                ;CPU ERROR
 7760 033710  022726  030000          4$:   CMP     #30000,(SP)+            ;IS PSW CORRECT ON STACK?
 7761 033714  001401                        BEQ     5$                      ;
 7762 033716  104001                        ERROR   .1                ;CPU ERROR
 7763 033720  005067  144042          5$:   CLR     CPEREG                  ;CLEAR CPU ERROR REG
 7764                                      ;
 7765                                      ;NOW WE'LL TRY TO FETCH AN INSTRUCTION FROM AN INTERNAL REGISTER.
 7766                                      ;THIS SHOULD CAUSE A TRAP TO ADDR 4 AND SET BIT 6 IN THE CPU
 7767                                      ;ERROR REGISTER.
 7768                                      ;
 7769 033724  012767  033752  144052        MOV     #7$,4                   ;LOAD VECTOR WITH TRAP HANDLER ADDR
 7770 033732  012767  030340  144046        MOV     #30340,6                ;LOAD VEC WITH PSW VALUE ON TRAP
 7771 033740  005067  144032                CLR     PS                      ;CLEAR THE PSW
 7772 033744  000167  144026                JMP     PS                      ;*******TEST INSTRUCTION********
 7773                                                                       ;TRY INSTRUCTION FETCH FROM INTERNAL
 7774                                                                       ;REGISTER-- SHOULD TRAP VIA ADDR 4
 7775 033750  104001                  6$:   ERROR   .1                ;CPU ERROR
 7776 033752  016701  144020          7$:   MOV     PS,R1                   ;SAVE CONTENTS OF PSW IN R1
 7777 033756  022767  000100  144002        CMP     #BIT06,CPEREG           ;IS CPU ERROR REGISTER CORRECT?
 7778 033764  001401                        BEQ     8$                      ;
 7779 033766  104001                        ERROR   .1                ;CPU ERROR
 7780 033770  022726  177776          8$:   CMP     #PS,(SP)+               ;IS STACK CONTENTS CORRECT?
 7781 033774  001401                        BEQ     9$                      ;
 7782 033776  104001                        ERROR   .1                ;CPU ERROR
 7783 034000  022726  000000          9$:   CMP     #0,(SP)+                ;IS STACK CONTENTS CORRECT?
 7784 034004  001401                        BEQ     10$                     ;
 7785 034006  104001                        ERROR   .1                ;CPU ERROR
 7786 034010  022701  000340          10$:  CMP     #340,R1                 ;WAS PSW LOADED PROPERLY ON TRAP?
 7787 034014  001401                        BEQ     11$                     ;
 7788 034016  104001                        ERROR   .1                ;CPU ERROR
 7789 034020  005067  143742          11$:  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
```

```
********** DOUBLE OPERAND TESTS *****************

 7790 034024  016767  146762  143752          MOV     SLOC00,4                ;RESTORE VECTOR
 7791 034032  016767  146756  143746          MOV     SLOC01,6                ;       "
 7792
 7795 034040                          RXXX:
 7796
 7797                                 ;       RED ZONE TRAP TEST
 7798 034040  013767  000004  146744          MOV     @#4,SLOC00              ;SAVE VECTOR
 7799 034046  012737  034076  000004          MOV     #2$,@#4                 ;SET UP VECTOR
 7800 034054  012706  000777                  MOV     #777,R6                 ;SET UP THE STACK WITH ODD ADDRESS.
 7801 034060  005067  143702                  CLR     CPEREG                  ;CLEAR THE CPU ERROR REGISTER
 7802 034064  005067  143706                  CLR     PSW                     ;CLEAR THE PSW
 7803 034070  005737  177700                  TST     @#177700                ;ACCESS NON-EXISTANT I/O ADDRESS
 7804 034074  104001                  1$:     ERROR   .1                ;CPU ERROR
 7805 034076  022706  000000          2$:     CMP     #0,SP                   ;IS R6 CORRECT?
 7806 034102  001401                          BEQ     3$                      ;BRANCH IF YES
 7807 034104  104001                          ERROR   .1                ;CPU ERROR
 7808 034106  022726  034074          3$:     CMP     #1$,(SP)+               ;IS DATA AT ADDR 0 CORRECT?
 7809 034112  001401                          BEQ     4$                      ;BRANCH IF YES
 7810 034114  104001                          ERROR   .1                ;CPU ERROR
 7811 034116  022716  000000          4$:     CMP     #0,(SP)                 ;IS PSW DATA IN ADDR 2 CORRECT?
 7812 034122  001401                          BEQ     5$                      ;BRANCH IF YES
 7813 034124  104001                          ERROR   .1                ;CPU ERROR
 7814 034126  022767  000124  143632  5$:     CMP     #124,CPEREG             ;IS CPU ERROR REGISTER CORRECT?
 7815 034134  001401                          BEQ     6$                      ;BRANCH IF YES
 7816 034136  104001                          ERROR   .1                ;CPU ERROR
 7817 034140  005067  143622          6$:     CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 7818 034144  012706  001000                  MOV     #STBOT,SP               ;RESTORE STACK
 7819 034150  016737  146636  000004          MOV     SLOC00,@#4              ;RESTORE VECTOR
 7820 034156  005037  000000                  CLR     @#0                     ;RESTORE ADDR 0
 7821 034162  005037  000002                  CLR     @#2                     ;RESTORE ADDR 2
 7822
 7824
 7826
 7827 034166                          PIRXXX:
 7828                                 ;       TEST PIRQ REGISTER RESPONSE
 7829 034166  016767  143612  146616          MOV     4,SLOC00                ;SAVE CONTENTS OF VECTORS
 7830 034174  012767  034222  143602          MOV     #PIRQNX,4               ;SETUP INTERRUPT VECTOR
 7831 034202  005067  143560                  CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 7832 034206  005737  177772                  TST     @#PIRQ                  ;LOOK FOR REPLY FROM PIRQ
 7833 034212  016767  146574  143564          MOV     SLOC00,4                ;RESTORE VECTORS
 7834 034220  000401                          BR      PIRTEX                  ;IF IT RESPONDS, CONTINUE TESTING.
 7835                                                                         ;ERROR! NO RESPONSE FROM PIRQ REG
 7836 034222  104001                  PIRQNX: ERROR   .1                ;CPU ERROR
 7837 034224                          PIRTEX:
 7838
 7839 034224                          PIR1:
 7840                                 ;       TEST PIRQ REGISTER DATA
 7841 034224  013767  000240  146560          MOV     @#PIRQVEC,SLOC00        ;SAVE PIRQ VECTORS
 7842 034232  013767  000242  146554          MOV     @#PIRQVEC+2,SLOC01      ;       "
 7843 034240  012737  034450  000240          MOV     #UNXPIR,@#PIRQVEC       ;SET UP PIRQ VECTOR FOR UNEXPECTED INTERRUPT
 7844 034246  012737  000340  000242          MOV     #PR7,@#PIRQVEC+2        ;
 7845 034254  012703  001000                  MOV     #1000,R3                ;PUT 1000 IN R3: START TESTING
 7846                                                                         ;BITS IN PIRQ REG BY FLOATING
 7847                                                                         ;A BIT THROUGH BITS 9-15.
 7848 034260  012704  034324                  MOV     #PIRTBL,R4              ;SET UP R4 AS A POINTER TO EXPECTED
 7849                                                                         ;ENCODED PRIORITY LEVELS IN PIRTBL
 7850 034264  000237                  1$:     SPL     7                       ;DON'T ALLOW INTERRUPTS.
```

```
********** DOUBLE OPERAND TESTS *****************

 7851 034266  005037  177772                CLR     @#PIRQ                  ;CLEAR OUT THE PIRQ
 7852 034272  023724  177772                CMP     @#PIRQ,(R4)+            ;IS PIRQ OK??
 7853 034276  001401                        BEQ     2$                      ;BRANCH IF OK
 7854                                                                       ;ERROR; PIRQ REG WAS NOT CLEAR
 7855 034300  104001                        ERROR   +1              ;CPU ERROR
 7856 034302  010337  177772          2$:   MOV     R3,@#PIRQ               ;SET A BIT IN PIRQ REGISTER
 7857 034306  023724  177772                CMP     @#PIRQ,(R4)+            ;COMPARE THE ENCODED PRIORITY BITS
 7858                                                                       ;WITH DATA IN THE TABLE (PIRTBL)
 7859 034312  001401                        BEQ     3$                      ;BRANCH IF ITS OK
 7860                                                                       ;ERROR; ENCODED PRIORITY LEVELS ARE
 7861                                                                       ;NOT CORRECT
 7862 034314  104001                        ERROR   +1              ;CPU ERROR
 7863 034316  006303                3$:     ASL     R3                      ;FLOAT A "1" THRU BITS 9-15
 7864 034320  103370                        BCC     2$                      ;IF CARRY BIT ISN'T SET, DO AGAIN.
 7865 034322  000410                        BR      EXPIR1                  ;GO TO EXIT TEST.
 7866
 7867 034324  000000          PIRTBL:       .WORD   0
 7868 034326  001042                        .WORD   1042
 7869 034330  002104                        .WORD   2104
 7870 034332  004146                        .WORD   4146
 7871 034334  010210                        .WORD   10210
 7872 034336  020252                        .WORD   20252
 7873 034340  040314                        .WORD   40314
 7874 034342  100356                        .WORD   100356
 7875
 7876 034344                    EXPIR1:
 7877
 7887
 7888 034344                    PIR2:
 7889                           ;       TEST PIRQ REGISTER LEVEL ENCODING
 7890                           ;;*****************************************************************
                                ;THIS TEST IS TO CHECK THAT THE HIGHEST PRIORITY LEVEL SET IN THE
                                ;PROGRAM INTERRUPT REQUEST BITS IS REFLECTED IN THE ENCODED PROGRAM
                                ;INTERRUPT ACTIVE BITS.
 7891 034344  000237                        SPL     7                       ;SHUT OFF INTERRUPTS
 7892 034346  005037  177772                CLR     @#PIRQ                  ;CLEAR PIRQ REGISTER
 7893 034352  012767  000001  146450        MOV     #1,FLAG                 ;SETUP END OF LOOP SIGNAL
 7894 034360  012703  177000                MOV     #177000,R3              ;SET UP DESIRED PATTERN IN R3
 7895 034364  012704  034430                MOV     #PITBL1,R4              ;SETUP R4 AS A TABLE POINTER
 7896 034370  010337  177772          1$:   MOV     R3,@#PIRQ               ;SET PATERN IN PIRQ REGISTER
 7897
 7898 034374  023724  177772                CMP     @#PIRQ,(R4)+            ;COMPARE PATTERN IN PIRQ WITH PATTERN IN
 7899                                                                       ;EXPECTED PATTERN TABLE.
 7900 034400  001012                        BNE     2$                      ;GO TO ERROR IF NOT THE SAME
 7901 034402  005737  003030                TST     @#FLAG                  ;IS THIS THE LAST TIME THROUGH??
 7902 034406  001421                        BEQ     PIR2EX                  ;YES, IF FLAG IS ZERO
 7903 034410  006003                        ROR     R3                      ;SHIFT PATTERN TO RIGHT FOR NEXT TEST
 7904 034412  042703  000777                BIC     #777,R3                 ;STRIP OFF BITS  8-0
 7905 034416  001364                        BNE     1$                      ;IF R3 IS NOT = 0 GO' DO THE NEXT PATTERN
 7906 034420  005037  003030                CLR     @#FLAG                  ;CLR THE FLAG TO INDICATE LAST
 7907                                                                       ;TIME THROUGH THE LOOP
 7908 034424  000761                        BR      1$                      ;GO THROUGH LOOP ONCE MORE
 7909                                                                       ;ERROR; PATTERNS WERE NOT THE SAME
 7910 034426  104001                2$:     ERROR   +1              ;CPU ERROR
 7911
 7912 034430  177356  077314  037252  PITBL1: .WORD   177356,77314,37252,17210,7146,3104,1042,0
 7913
```

```
********** DOUBLE OPERAND TESTS ******************

7914 034450                          UNXPIR:                                ;UNEXPECTED PIRQ INTERRUPT
7915 034450  104001                          ERROR   +1              ;CPU ERROR
7916
7917 034452                          PIR2EX:
7925
7926 034452                          PIR3:
7927                                 ;       TEST PIRQ INTERRUPTS
7928                                 ;;*************************************************************************
                                     ;THIS TEST CHECKS THAT EACH PROGRAM INTERRUPT OCCURS PROPERLY
7929 034452  012703  001000                  MOV     #1000,R3                ;SETUP R3 AS A WORKING REGISTER
7930 034456  012737  034522  000240          MOV     #PIRRTN,@#PIRQVEC       ;SET UP INTERRUPT ROUTINE AT PIR VECTOR
7931 034464  012737  000340  000242          MOV     #340,@#PIRQVEC+2        ;       "       "       "       "
7932 034472  005004                          CLR     R4                      ;INITIALIZE R4 AS EXPECTED DATA HOLDER.
7933 034474  005724                  PI1:    TST     (R4)+                   ;INCREMENT R4 BY 2
7934 034476  050337  177772                  BIS     R3,@#PIRQ               ;SET PIRQ TO INTERRUPT
7935 034502  000230                          SPL     0                       ;ENABLE INTERRUPT TO OCCUR
7936                                                                         ;ERROR! INTERRUPT DID NOT OCCUR
7937 034504  104001                          ERROR   +1              ;CPU ERROR
7938                                                                         ;ERROR! INTERRUPT OCCURRED IN WRONG SEQUENCE
7939 034506  104001                  PI2:    ERROR   +1              ;CPU ERROR
7940 034510  062706  000004          PI3:    ADD     #4,SP                   ;CLEAN UP THE STACK
7941 034514  006303                          ASL     R3                      ;SHIFT R3 TO LEFT FOR NEXT INTERRUPT
7942 034516  103366                          BCC     PI1                     ;END IF CARRY BIT IS SET
7943 034520  000412                          BR      PIR3EX                  ;ON TO THE NEXT TEST
7944
7945 034522  013705  177772          PIRRTN: MOV     @#PIRQ,R5               ;MOVE THE CONTENTS OF PIRQ REG TO R5
7946 034526  005037  177772                  CLR     @#PIRQ                  ;KILL PIRQ INTERRUPT.
7947 034532  042705  177761                  BIC     #177761,R5              ;MASK OFF ALL BITS EXCEPT 1-4.
7948 034536  020405                          CMP     R4,R5                   ;DID THE CORRECT LEVEL INTERRUPT OCCUR?
7949 034540  001362                          BNE     PI2                     ;IF NOT; GO TO ERROR.
7950 034542  000167  177742                  JMP     PI3                     ;RETURN FROM SUCCESSFUL INTERRUPT,GET
7951                                                                         ;READY FOR THE NEXT ONE.
7952
7953 034546                          PIR3EX:
7954
7963
7964 034546                          PIR4:
7965                                 ;       TEST PIRQ VS PSW INTERRUPT LEVEL
7966                                 ;;*************************************************************************
                                     ;THIS TEST IS TO ENSURE THAT PIRQ CANNOT INTERRUPT WHEN PSW INTERRUPT
                                     ;LEVEL IS SET TO BLOCK THEM OUT.
7967 034546  005037  177772                  CLR     @#PIRQ                  ;CLEAR THE PIRQ
7968 034552  005037  177776                  CLR     @#PSW                   ;CLEAR THE PSW.
7969 034556  000237                          SPL     7                       ;BLOCK INTERRUPTS FROM BEING SERVICED.
7970 034560  012703  001000                  MOV     #1000,R3                ;USE R3 AS A WORKING REGISTER.
7971 034564  012737  034616  000240          MOV     #2$,@#PIRQVEC           ;SETUP INTERRUPT VECTORS
7972 034572  012737  000340  000242          MOV     #340,@#PIRQVEC+2                ;
7973 034600  010337  177772          1$:     MOV     R3,@#PIRQ               ;SET INTERRUPT LEVEL IN PIRQ.
7974 034604  006303                          ASL     R3                      ;SHIFT A "1" LEFT TO INCREASE INTERRUPT
7975                                                                         ;PRIORITY LEVEL.
7976 034606  103374                          BCC     1$                      ;IF C BIT NOT SET THEN DO IT AGAIN.
7977 034610  005037  177772                  CLR     @#PIRQ                  ;ELSE CLEAR THE PIRQ
7978 034614  000403                          BR      3$                      ;EXIT TEST.
7979
7980 034616  005037  177772          2$:     CLR     @#PIRQ                  ;STOP PIRQ FROM INTERRUPTING.
7981                                                                         ;ERROR! NO INTERRUPTS SHOULD OCCUR.
7982 034622  104001                          ERROR   +1              ;CPU ERROR
```

```
********** DOUBLE OPERAND TESTS *****************

 7983 034624                            3$:
 7984
 7998
 7999 034624                            PIR5:
 8000                                   ;       TEST PIRQ INTERRUPTS OCCUR AT PROPER LEVEL
 8001                                   ;;*****************************************************************
                                        ;THIS TEST ENSURES THAT INTERRUPTS OCCUR AT THE PROPER LEVEL
                                        ;THE PRIORITY LEVEL IS INITIALLY SET TO PRI6. THE PIRQ THEN STARTS INTERRUPTING
                                        ;AT PRI1. NO INTERRUPTS SHOULD OCCUR UNTIL THE INTERRUPTING LEVEL EXCEEDS THE
                                        ;PRIORITY LEVEL IN THE PSW. AFTER EACH LEVEL OF INTERRUPT HAS BEEN TRIED; THE
                                        ;PSW PRIORITY LEVEL IS LOWERED ONE NOTCH, AND THE CYCLE REPEATS UNTIL PSW
                                        ;PRIORITY LEVEL 0 IS REACHED. INTERRUPTS AT PSW PRIORITY LEVEL 0 HAVE BEEN DONE
                                        ;PREVIOUSLY.
 8002 034624  005037  177772                    CLR     @#PIRQ                  ;CLR THE PIRQ
 8003 034630  012737  034742  000240            MOV     #10$,@#PIRQVEC          ;SET UP INTERRUPT VECTORS
 8004 034636  012737  000340  000242            MOV     #PR7,@#PIRQVEC+2        ;
 8005 034644  012705  035020                    MOV     #PITBL2,R5              ;SETUP R5 AS A TABLE POINTER.
 8006 034650  012737  000006  114144            MOV     #6,@#DCOUNT             ;INIT LOCATION DCOUNT:
 8007 034656  005004                    1$:     CLR     R4                      ;INITIALIZE R4 AS A COUNTER.
 8008 034660  012703  001000                    MOV     #1000,R3                ;USE R3 AS A WORKING REGISTER.
 8009 034664  012567  143106                    MOV     (R5)+,PS                ;MOVE PRIORITY LEVEL TO PSW
 8010 034670  005724                    3$:     TST     (R4)+                   ;INCREMENT R4 BY 2
 8011 034672  010337  177772                    MOV     R3,@#PIRQ               ;SET INTERRUPT LEVEL IN PIRQ.
 8012                                   ;*******INTERRUPTS WILL HAPPEN HERE*********
 8013 034676  013701  177776                    MOV     @#PS,R1                 ;SAVE PS IN R1          !ONLY EXECUTED
 8014 034702  013702  177772                    MOV     @#PIRQ,R2               ;SAVE PIRQ IN R2        !   IF NO
 8015 034706  042701  177437                    BIC     #177437,R1              ;CLEAR EXTRANEOUS BITS  ! INTERRUPT
 8016 034712  042702  177437                    BIC     #177437,R2              ;                       !    HAS
 8017 034716  020102                            CMP     R1,R2                   ;R1 SHOULD BE >= R2     !  OCCURRED
 8018 034720  002001                            BGE     4$                      ;IF IT IS GO ON         !-------
 8019                                                                           ;ERROR! SHOULD HAVE INTERRUPTED.
 8020 034722  104001                            ERROR   +1                  ;CPU ERROR
 8021 034724  006303                    4$:     ASL     R3                      ;SHIFT A "1" LEFT UNTIL INTERRUPT LEVEL
 8022                                                                           ;IS REACHED.
 8023 034726  103360                            BCC     3$                      ;BRANCH BACK IF CARRY IS NOT SET.
 8024 034730  005337  114144                    DEC     @#DCOUNT                ;LOWER INTERRUPT PRIORITY LEVEL
 8025 034734  001350                            BNE     1$                      ;IF DCOUNT= 0 , WE'RE DONE.
 8026 034736  000167  000072                    JMP     PIR5EX                  ;JUMP TO END OF THIS TEST.
 8027
 8028                                   ;PIRQ INTERRUPT SERVICE ROUTINE
 8029                                   ;
 8030
 8031 034742  013746  177772            10$:    MOV     @#PIRQ,-(SP)            ;SAVE PIRQ DATA ON STACK
 8032 034746  005037  177772                    CLR     @#PIRQ                  ;SHUT OFF PIRQ INTERRUPTS
 8033 034752  016601  000004                    MOV     4(SP),R1                ;GET OLD PSW. SAVE IN R1.
 8034 034756  011602                            MOV     (SP),R2                 ;GET PIRQ FROM STACK.
 8035 034760  042702  177437                    BIC     #177437,R2              ;CLEAR UNWANTED BITS FROM R2
 8036 034764  042701  177437                    BIC     #177437,R1              ;CLEAR UNWANTED BITS FROM R1
 8037 034770  020102                            CMP     R1,R2                   ;R2 SHOULD BE > R1.
 8038 034772  100401                            BMI     20$                     ;GO CHECK SEQUENCE OF INTERRUPT.
 8039                                                                           ;ERROR! PRIORITY OF INTERRUPT WAS NOT
 8040                                                                           ;HIGH ENOUGH. SHOULDN'T HAVE OCCURRED.
 8041 034774  104001                            ERROR   +1                  ;CPU ERROR
 8042 034776  012602                    20$:    MOV     (SP)+,R2                ;POP OLD PIRQ OFF THE STACK.
 8043 035000  042702  177761                    BIC     #177761,R2              ;CLEAR OFF EXTRANEOUS BITS.
 8044 035004  020402                            CMP     R4,R2                   ;SHOULD BE EQUAL.
 8045 035006  001401                            BEQ     21$                     ;IF THEY ARE EQUAL, CLEAN UP STACK AND
```

```
********** DOUBLE OPERAND TESTS *****************

                                                                ;GET READY FOR THE NEXT INTERRUPT
                                                                ;ELSE
                                                                ;ERROR! INTERRUPT OCCURRED OUT OF SEQUENCE.
035010  104001                  ERROR   +1              ;CPU ERROR
035012  012716  034724  21$:    MOV     #4$,(SP)                ;PUT RETURN ADDRESS ON THE STACK.
035016  000002                  RTI                             ;RETURN FROM INTERRUPT. RESTORE PSW.

035020  000300          PITBL2: .WORD   300             ;PRIORITY LEVEL 6
035022  000240                  .WORD   240             ;PRIORITY LEVEL 5
035024  000200                  .WORD   200             ;PRIORITY LEVEL 4
035026  000140                  .WORD   140             ;PRIORITY LEVEL 3
035030  000100                  .WORD   100             ;PRIORITY LEVEL 2
035032  000040                  .WORD   40              ;PRIORITY LEVEL 1

035034                  PIR5EX:


035034                  PIR6:
                        ;       TEST THAT PIRQS ARE SERVICED IN CORRECT ORDER
                        ;;*****************************************************************
                        ;THIS TEST CHECKS THAT ALL PIRQ INTERRUPTS ARE SERVICED
                        ;IN THE CORRECT DECENDING ORDER. I.E. IRQ6 IS NOT SERVICED
                        ;BEFORE IRQ7 ETC. THE PIRQ IS LOADED WITH ALL INTERRUPTS SIMULTANEOUSLY
                        ;TURNED ON.  THE PSW PRIORITY LEVEL IS THEN LOWERED TO 0. EACH INTERRUPT
                        ;SHOULD BE SERVICED IN DECENDING ORDER. EACH TIME AN INTERRUPT OCCURRS, THE PSW
                        ;IS LOADED WITH PRIORITY LEVEL 7 WHICH STOPS THE PIRQ FROM FURTHER INTERRUPTS.
                        ;AFTER A CORRECT INTERRUPT, AN RTI IS EXECUTED, AND THE PSW PRIORITY LEVEL
                        ;RETRUNS TO ZERO ALLOWING THE NEXT LOWER INTERRUPT TO OCCUR.
035034  005037  177772          CLR     @#PIRQ                  ;CLEAR OUT THE PIRQ
035040  000237                  SPL     7                       ;NO INTERRUPTS WILL BE SERVICED.
035042  012737  035102  000240  MOV     #10$,@#PIRQVEC          ;SET UP VECTORS
035050  012737  000340  000242  MOV     #PR7,@#PIRQVEC+2        ;
035056  012703  177000          MOV     #177000,R3              ;SETUP DESIRED PATTERN IN R3
035062  010337  177772          MOV     R3,@#PIRQ               ;MOVE PATTERN IN R3 TO PIRQ.
035066  012704  000016          MOV     #16,R4                  ;PRELOAD R4 AS A DECREMENTING COUNTER.
035072  000230                  SPL     0                       ;LET THE PIRQ INTERRUPT
                        ;*********INTERRUPTS OCCUR HERE!!!*********************
035074  005704                  TST     R4                      ;IS R4 ZERO? IT SHOULD BE. THIS
                                                                ;INSTRUCTION SHOULDN'T BE EXECUTED
                                                                ;UNTIL ALL INTERRUPTS HAVE OCCURRED.
035076  001417                  BEQ     PIR6EX                  ;GO TO NEXT TEST.
                                                                ;ERROR! ALL INTERRUPTS DID NOT OCCUR.
035100  104001                  ERROR   +1              ;CPU ERROR
035102  013702  177772  10$:    MOV     @#PIRQ,R2               ;SAVE THE PIRQ IN R2
035106  042702  177761          BIC     #177761,R2              ;STRIP OFF EXTRANEOUS BITS.
035112  020402                  CMP     R4,R2                   ;SHOULD BE EQUAL.
035114  001401                  BEQ     15$                     ;SETUP FOR NEXT INTERRUPT.
                                                                ;ERROR! INTERRUPT WAS SERVICED OUT OF ORDER
035116  104001                  ERROR   +1              ;CPU ERROR
035120  005744          15$:    TST     -(R4)                   ;DECREMENT R4 BY 2
035122  006003                  ROR     R3                      ;CHANGE PATTERN IN R3; LOWER INTERRUPT PRIORITY
035124  042703  000777          BIC     #777,R3                 ;STRIP OFF UNNEEDED BITS
035130  010337  177772          MOV     R3,@#PIRQ               ;LOWER INTERRUPT PRIORITY LEVEL.
035134  000002                  RTI                             ;

035136  016737  145650  000240  PIR6EX: MOV     SLOC00,@#PIRQVEC        ;RESTORE VECTORS
035144  016737  145644  000242          MOV     SLOC01,@#PIRQVEC+2              ;       "
```

```
********** DOUBLE OPERAND TESTS *****************

 8109                                                    ;
 8111
 8114                                .SBTTL  MEMORY MANAGEMENT TESTS
 8115                                ;*************************************************************************
 8116                                ;*************************************************************************
 8117                                ;                       BEGIN MMU TESTING
 8118                                ;*************************************************************************
 8119                                ;*************************************************************************
 8121 035152                        TSMMU1:
 8122                                ;       STATUS REGISTER TEST
 8123 035152  005067  142610                    CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 8124 035156  005037  177572                    CLR     @#177572                ;TURN MMU OFF
 8125 035162  005037  003030                    CLR     @#FLAG                  ;CLEAR MMU TRAP FLAG
 8126 035166  013746  000004                    MOV     @#4,-(SP)               ;SAVE OLD VECTOR
 8127 035172  012737  137720  000004           MOV     #ADDTRP,@#4             ;SETUP NEW VECTOR
 8128 035200  005005                            CLR     R5                      ;CLEAR FLAG
 8129 035202  013701  177572                    MOV     @#177572,R1             ; TEST MMR0
 8130 035206  013701  177574                    MOV     @#177574,R1             ; TEST MMR1
 8131 035212  013701  177576                    MOV     @#177576,R1             ; TEST MMR2
 8132 035216  013701  172516                    MOV     @#172516,R1             ; TEST MMR3
 8133 035222  012637  000004                    MOV     (SP)+,@#4               ;RESTORE VECTOR
 8134 035226  020527  000000                    CMP     R5,#0                   ;DID WE TRAP
 8135 035232  001401                            BEQ     1$                      ;NO, THEN BRANCH
 8136 035234  104002                            ERROR   +2              ;MMU ERROR
 8137                                                                   ;YES, GO TO ERROR
 8138 035236                        1$:
 8141 035236                        TSMMU2:
 8142                                ;       ADDRESS TEST OF PARS,PDRS, AND FP REGS
 8143 035236  005067  142524                    CLR     CPEREG                  ;CLEAR CPU ERROR REGISTER
 8144 035242  005037  177572                    CLR     @#177572                ;MMU OFF
 8145 035246  005037  003030                    CLR     @#FLAG                  ;CLEAR MMU TRAP FLAG
 8146 035252  013746  000244                    MOV     @#244,-(SP)             ;SAVE FP VECTOR
 8147 035256  013746  000246                    MOV     @#246,-(SP)
 8148 035262  013746  000004                    MOV     @#4,-(SP)               ;SAVE TIME OUT VECTOR
 8149 035266  012737  000246  000244           MOV     #246,@#244              ;SETUP NEW FP VECTOR
 8150 035274  012737  000002  000246           MOV     #2,@#246                ;
 8151 035302  012737  137720  000004           MOV     #ADDTRP,@#4             ;SETUP NEW TIME OUT VECTOR
 8152 035310  005005                            CLR     R5                      ;CLEAR TIMEOUT FLAG
 8153 035312  012700  172200                    MOV     #172200,R0              ;LOAD ALL PARS AND PDRS WITH ZERO
 8154 035316  005020                    1$:     CLR     (R0)+                   ;
 8155 035320  020027  172400                    CMP     R0,#172400              ;
 8156 035324  001374                            BNE     1$                      ;
 8157 035326  012700  177600                    MOV     #177600,R0              ;
 8158 035332  005020                    2$:     CLR     (R0)+                   ;
 8159 035334  020027  177700                    CMP     R0,#177700              ;
 8160 035340  001374                            BNE     2$                      ;
 8161 035342  170127  000200                    LDFPS   #200                    ;
 8162 035346  012700  003052                    MOV     #FLOAT,R0               ;LOAD AC0-AC5 WITH 0
 8163 035352  005020                            CLR     (R0)+                   ;
 8164 035354  005020                            CLR     (R0)+                   ;
 8165 035356  005020                            CLR     (R0)+                   ;
 8166 035360  005020                            CLR     (R0)+                   ;
 8167 035362  012700  003052                    MOV     #FLOAT,R0               ;
 8168 035366  172410                            LDD     (R0),AC0                ;
 8169 035370  172510                            LDD     (R0),AC1                ;
 8170 035372  172610                            LDD     (R0),AC2                ;
 8171 035374  172710                            LDD     (R0),AC3                ;
```

```
8172 035376  174004                    STD     AC0,AC4                 ;
8173 035400  174005                    STD     AC0,AC5                 ;
8174 035402  174500              3$:   DIVD    AC0,AC1                 ;LOAD FEC WITH 4 AND FEA WITH #3$
8175 035404  170337  003062            STST    @#FLO                   ;CHECK FEC FOR 4 AND FEA FOR #3$
8176 035410  012704  003062            MOV     #FLO,R4                 ;
8177 035414  022427  000004            CMP     (R4)+,#4                ;
8178 035420  001401                    BEQ     21$                     ;
8179 035422  104002                    ERROR   +2              ;MMU ERROR
8180                                                                   ;
8181 035424  021427  035402      21$:  CMP     (R4),#3$                ;
8182 035430  001401                    BEQ     22$                     ;
8183 035432  104002                    ERROR   +2              ;MMU ERROR
8184                                                                   ;
8185 035434  012704  172200      22$:  MOV     #172200,R4              ;CHECK EACH PAR, PDR FOR 0 THEN
8186 035440  012701  000001            MOV     #1,R1                   ;WRITE A UNIQUE NUMBER TO IT
8187 035444  010102              4$:   MOV     R1,R2                   ;
8188 035446  072227  000010            ASH     #10,R2                  ;
8189 035452  021427  000000            CMP     (R4),#0                 ;
8190 035456  001401                    BEQ     5$                      ;
8191 035460  104002                    ERROR   +2              ;MMU ERROR
8192                                                                   ;
8193 035462  010224              5$:   MOV     R2,(R4)+                ;
8194 035464  005201                    INC     R1                      ;
8195 035466  020427  172400            CMP     R4,#172400              ;
8196 035472  001364                    BNE     4$                      ;
8197 035474  012704  177600            MOV     #177600,R4              ;
8198 035500  010102              6$:   MOV     R1,R2                   ;
8199 035502  072227  000010            ASH     #10,R2                  ;
8200 035506  021427  000000            CMP     (R4),#0                 ;
8201 035512  001401                    BEQ     7$                      ;
8202 035514  104002                    ERROR   +2              ;MMU ERROR
8203                                                                   ;
8204 035516  010224              7$:   MOV     R2,(R4)+                ;
8205 035520  005201                    INC     R1                      ;
8206 035522  020427  177700            CMP     R4,#177700              ;
8207 035526  001364                    BNE     6$                      ;
8208 035530  012704  003062            MOV     #FLO,R4                 ;CHECK AC5 FOR ALL ZEROES THEN LOAD A 6
8209 035534  012703  003052            MOV     #FLOAT,R3               ;
8210 035540  174014                    STD     AC0,(R4)                ;
8211 035542  172405                    LDD     AC5,AC0                 ;
8212 035544  174013                    STD     AC0,(R3)                ;
8213 035546  012702  000004            MOV     #4,R2                   ;
8214 035552  022327  000000      8$:   CMP     (R3)+,#0                ;
8215 035556  001401                    BEQ     9$                      ;
8216 035560  104002                    ERROR   +2              ;MMU ERROR
8217                                                                   ;
8218 035562  005302              9$:   DEC     R2                      ;
8219 035564  001372                    BNE     8$                      ;
8220 035566  012703  003052            MOV     #FLOAT,R3               ;
8221 035572  012713  000006            MOV     #6,(R3)                 ;
8222 035576  172413                    LDD     (R3),AC0                ;
8223 035600  174005                    STD     AC0,AC5                 ;
8224 035602  172404                    LDD     AC4,AC0                 ;CHECK AC4 FOR ALL ZEROES THEN LOAD A 5
8225 035604  174013                    STD     AC0,(R3)                ;
8226 035606  012702  000004            MOV     #4,R2                   ;
8227 035612  022327  000000      10$:  CMP     (R3)+,#0                ;
8228 035616  001401                    BEQ     11$                     ;
```

MEMORY MANAGEMENT TESTS

```
8229 035620  104002                        ERROR   +2              ;MMU ERROR
8230                                                               ;
8231 035622  005302              11$:      DEC     R2                      ;
8232 035624  001372                        BNE     10$                     ;
8233 035626  012703  003052                MOV     #FLOAT,R3               ;
8234 035632  012713  000005                MOV     #5,(R3)                 ;
8235 035636  172413                        LDD     (R3),AC0                ;
8236 035640  174004                        STD     AC0,AC4                 ;
8237 035642  012702  000004                MOV     #4,R2                   ;CHECK AC0 FOR ALL ZEROES THEN LOAD A 1
8238 035646  022427  000000        12$:    CMP     (R4)+,#0                ;
8239 035652  001401                        BEQ     13$                     ;
8240 035654  104002                        ERROR   +2              ;MMU ERROR
8241                                                               ;
8242 035656  005302              13$:      DEC     R2                      ;
8243 035660  001372                        BNE     12$                     ;
8244 035662  012713  000001                MOV     #1,(R3)                 ;
8245 035666  172413                        LDD     (R3),AC0                ;
8246 035670  012704  003062                MOV     #FLO,R4                 ;CHECK AC1 FOR ALL ZEROES THEN LOAD A 2
8247 035674  012702  000004                MOV     #4,R2                   ;
8248 035700  174114                        STD     AC1,(R4)                ;
8249 035702  022427  000000        14$:    CMP     (R4)+,#0                ;
8250 035706  001401                        BEQ     15$                     ;
8251 035710  104002                        ERROR   +2              ;MMU ERROR
8252                                                               ;
8253 035712  005302              15$:      DEC     R2                      ;
8254 035714  001372                        BNE     14$                     ;
8255 035716  012713  000002                MOV     #2,(R3)                 ;
8256 035722  172513                        LDD     (R3),AC1                ;
8257 035724  012704  003062                MOV     #FLO,R4                 ;CHECK AC2 FOR ALL ZEROES THEN LOAD A 3
8258 035730  012702  000004                MOV     #4,R2                   ;
8259 035734  174214                        STD     AC2,(R4)                ;
8260 035736  022427  000000        16$:    CMP     (R4)+,#0                ;
8261 035742  001401                        BEQ     17$                     ;
8262 035744  104002                        ERROR   +2              ;MMU ERROR
8263                                                               ;
8264 035746  005302              17$:      DEC     R2                      ;
8265 035750  001372                        BNE     16$
8266 035752  012713  000003                MOV     #3,(R3)                 ;
8267 035756  172613                        LDD     (R3),AC2                ;
8268 035760  012704  003062                MOV     #FLO,R4                 ;CHECK AC3 FOR ALL ZEROES THEN LOAD A 4
8269 035764  012702  000004                MOV     #4,R2                   ;
8270 035770  174314                        STD     AC3,(R4)                ;
8271 035772  022427  000000        18$:    CMP     (R4)+,#0                ;
8272 035776  001401                        BEQ     19$                     ;
8273 036000  104002                        ERROR   +2              ;MMU ERROR
8274                                                               ;
8275 036002  005302              19$:      DEC     R2                      ;
8276 036004  001372                        BNE     18$                     ;
8277 036006  012713  000004                MOV     #4,(R3)                 ;
8278 036012  172713                        LDD     (R3),AC3                ;
8279 036014  012704  003062                MOV     #FLO,R4                 ;CHECK FPS FOR 100204 THEN LOAD IT WITH 200
8280 036020  170214                        STFPS   (R4)                    ;
8281 036022  022714  100204                CMP     #100204,(R4)            ;
8282 036026  001401                        BEQ     20$                     ;
8283 036030  104002                        ERROR   +2              ;MMU ERROR
8284                                                               ;
8285 036032  170127  000200        20$:    LDFPS   #200                    ;
```

MEMORY MANAGEMENT TESTS

```
8286 036036  012704  172200              MOV     #172200,R4              ;CHECK PDR, PAR FOR UNIQUE NUMBERS
8287 036042  012701  000001              MOV     #1,R1                   ;
8288 036046  010102              23$:    MOV     R1,R2                   ;
8289 036050  072227  000010              ASH     #10,R2                  ;
8290 036054  022402                      CMP     (R4)+,R2                ;
8291 036056  001401                      BEQ     24$                     ;
8292 036060  104002                      ERROR   +2              ;MMU ERROR
8293                                                             ;
8294 036062  005201              24$:    INC     R1                      ;
8295 036064  020427  172400              CMP     R4,#172400              ;
8296 036070  001366                      BNE     23$                     ;
8297 036072  012704  177600              MOV     #177600,R4              ;
8298 036076  010102              25$:    MOV     R1,R2                   ;
8299 036100  072227  000010              ASH     #10,R2                  ;
8300 036104  022402                      CMP     (R4)+,R2                ;
8301 036106  001401                      BEQ     26$                     ;
8302 036110  104002                      ERROR   +2              ;MMU ERROR
8303                                                             ;
8304 036112  005201              26$:    INC     R1                      ;
8305 036114  020427  177700              CMP     R4,#177700              ;
8306 036120  001366                      BNE     25$                     ;
8307 036122  012701  003052              MOV     #FLOAT,R1               ;CHECK AC5 FOR #6
8308 036126  012704  003062              MOV     #FLO,R4                 ;
8309 036132  174014                      STD     AC0,(R4)                ;
8310 036134  172405                      LDD     AC5,AC0                 ;
8311 036136  174011                      STD     AC0,(R1)                ;
8312 036140  022127  000006              CMP     (R1)+,#6                ;
8313 036144  001401                      BEQ     27$                     ;
8314 036146  104002                      ERROR   +2              ;MMU ERROR
8315                                                             ;
8316 036150  012703  000003      27$:    MOV     #3,R3                   ;
8317 036154  022127  000000      28$:    CMP     (R1)+,#0                ;
8318 036160  001401                      BEQ     29$                     ;
8319 036162  104002                      ERROR   +2              ;MMU ERROR
8320                                                             ;
8321 036164  005303              29$:    DEC     R3                      ;
8322 036166  001372                      BNE     28$                     ;
8323 036170  012701  003052              MOV     #FLOAT,R1               ;CHECK AC4 FOR #5
8324 036174  172404                      LDD     AC4,AC0                 ;
8325 036176  174011                      STD     AC0,(R1)                ;
8326 036200  022127  000005              CMP     (R1)+,#5                ;
8327 036204  001401                      BEQ     30$                     ;
8328 036206  104002                      ERROR   +2              ;MMU ERROR
8329                                                             ;
8330 036210  012703  000003      30$:    MOV     #3,R3                   ;
8331 036214  022127  000000      31$:    CMP     (R1)+,#0                ;
8332 036220  001401                      BEQ     32$                     ;
8333 036222  104002                      ERROR   +2              ;MMU ERROR
8334                                                             ;
8335 036224  005303              32$:    DEC     R3                      ;
8336 036226  001372                      BNE     31$                     ;
8337 036230  022427  000001              CMP     (R4)+,#1                ;CHECK AC0 FOR #1
8338 036234  001401                      BEQ     33$                     ;
8339 036236  104002                      ERROR   +2              ;MMU ERROR
8340                                                             ;
8341 036240  012703  000003      33$:    MOV     #3,R3                   ;
8342 036244  022427  000000      34$:    CMP     (R4)+,#0                ;
```

MEMORY MANAGEMENT TESTS

```
 8343 036250  001401                        BEQ     35$                 ;
 8344 036252  104002                        ERROR   +2                  ;MMU ERROR
 8345                                                                   ;
 8346 036254  005303                35$:     DEC     R3                  ;
 8347 036256  001372                        BNE     34$                 ;
 8348 036260  012701  003052                MOV     #FLOAT,R1           ;CHECK AC1 FOR #2
 8349 036264  174111                        STD     AC1,(R1)            ;
 8350 036266  022127  000002                CMP     (R1)+,#2            ;
 8351 036272  001401                        BEQ     36$                 ;
 8352 036274  104002                        ERROR   +2                  ;MMU ERROR
 8353                                                                   ;
 8354 036276  012703  000003        36$:    MOV     #3,R3               ;
 8355 036302  022127  000000        37$:    CMP     (R1)+,#0            ;
 8356 036306  001401                        BEQ     38$                 ;
 8357 036310  104002                        ERROR   +2                  ;MMU ERROR
 8358                                                                   ;
 8359 036312  005303                38$:    DEC     R3                  ;
 8360 036314  001372                        BNE     37$                 ;
 8361 036316  012701  003052                MOV     #FLOAT,R1           ;CHECK AC2 FOR #3
 8362 036322  174211                        STD     AC2,(R1)            ;
 8363 036324  022127  000003                CMP     (R1)+,#3            ;
 8364 036330  001401                        BEQ     39$                 ;
 8365 036332  104002                        ERROR   +2                  ;MMU ERROR
 8366                                                                   ;
 8367 036334  012703  000003        39$:    MOV     #3,R3               ;
 8368 036340  022127  000000        40$:    CMP     (R1)+,#0            ;
 8369 036344  001401                        BEQ     41$                 ;
 8370 036346  104002                        ERROR   +2                  ;MMU ERROR
 8371                                                                   ;
 8372 036350  005303                41$:    DEC     R3                  ;
 8373 036352  001372                        BNE     40$                 ;
 8374 036354  012701  003052                MOV     #FLOAT,R1           ;CHECK AC3 FOR #4
 8375 036360  174311                        STD     AC3,(R1)            ;
 8376 036362  022127  000004                CMP     (R1)+,#4            ;
 8377 036366  001401                        BEQ     42$                 ;
 8378 036370  104002                        ERROR   +2                  ;MMU ERROR
 8379                                                                   ;
 8380 036372  012703  000003        42$:    MOV     #3,R3               ;
 8381 036376  022127  000000        43$:    CMP     (R1)+,#0            ;
 8382 036402  001401                        BEQ     44$                 ;
 8383 036404  104002                        ERROR   +2                  ;MMU ERROR
 8384                                                                   ;
 8385 036406  005303                44$:    DEC     R3                  ;
 8386 036410  001372                        BNE     43$                 ;
 8387 036412  020527  000000                CMP     R5,#0               ;IS TIME OUT FLAG 0
 8388 036416  001401                        BEQ     45$                 ;YES GO ON
 8389 036420  104002                        ERROR   +2                  ;MMU ERROR
 8390                                                                   ;NO GO TO ERROR
 8391 036422  012637  000004        45$:    MOV     (SP)+,@#4           ;RESTORE TIME OUT VECTOR
 8392 036426  012637  000246                MOV     (SP)+,@#246         ;RESTORE FP VECTOR
 8393 036432  012637  000244                MOV     (SP)+,@#244         ;
 8394
 8397 036436                        TSMMU3:
 8398                               ;       WRITE ALL PARS/PDRS WITH ONES THEN ZEROS
 8399 036436  005037  177572                CLR     @#177572            ;MMU OFF
 8400 036442  005037  003030                CLR     @#FLAG              ;CLEAR MMU ABORT FLAG
 8401 036446  012703  172200                MOV     #172200,R3          ;LOAD ALL PARS AND PDRS WITH ONES
```

MEMORY MANAGEMENT TESTS

```
8402 036452  012723  177777          1$:     MOV     #177777,(R3)+           ;
8403 036456  020327  172400                  CMP     R3,#172400              ;
8404 036462  001373                          BNE     1$                      ;
8405 036464  012703  177600                  MOV     #177600,R3              ;
8406 036470  012723  177777          2$:     MOV     #177777,(R3)+           ;
8407 036474  020327  177700                  CMP     R3,#177700              ;
8408 036500  001373                          BNE     2$                      ;
8409 036502  012703  172200                  MOV     #172200,R3              ;CHECK SPDRS FOR ONES
8410 036506  022327  177416          3$:     CMP     (R3)+,#177416           ;
8411 036512  001401                          BEQ     4$                      ;
8412 036514  104002                          ERROR   +2              ;MMU ERROR
8413                                                                 ;
8414 036516  020327  172240          4$:     CMP     R3,#172240              ;
8415 036522  001371                          BNE     3$                      ;
8416 036524  022327  177777          5$:     CMP     (R3)+,#177777           ;CHECK SPARS FOR ONES
8417 036530  001401                          BEQ     6$                      ;
8418 036532  104002                          ERROR   +2              ;MMU ERROR
8419                                                                 ;
8420 036534  020327  172300          6$:     CMP     R3,#172300              ;
8421 036540  001371                          BNE     5$                      ;
8422 036542  022327  177416          7$:     CMP     (R3)+,#177416           ;CHECK KPDRS FOR ONES
8423 036546  001401                          BEQ     8$                      ;
8424 036550  104002                          ERROR   +2              ;MMU ERROR
8425                                                                 ;
8426 036552  020327  172340          8$:     CMP     R3,#172340              ;
8427 036556  001371                          BNE     7$                      ;
8428 036560  022327  177777          9$:     CMP     (R3)+,#177777           ;CHECK KPARS FOR ONES
8429 036564  001401                          BEQ     10$                     ;
8430 036566  104002                          ERROR   +2              ;MMU ERROR
8431                                                                 ;
8432 036570  020327  172400          10$:    CMP     R3,#172400              ;
8433 036574  001371                          BNE     9$                      ;
8434 036576  012703  177600                  MOV     #177600,R3              ;CHECK UPDRS FOR ONES
8435 036602  022327  177416          11$:    CMP     (R3)+,#177416           ;
8436 036606  001401                          BEQ     12$                     ;
8437 036610  104002                          ERROR   +2              ;MMU ERROR
8438                                                                 ;
8439 036612  020327  177640          12$:    CMP     R3,#177640              ;
8440 036616  001371                          BNE     11$                     ;
8441 036620  022327  177777          13$:    CMP     (R3)+,#177777           ;CHECK UPARS FOR ONES
8442 036624  001401                          BEQ     14$                     ;
8443 036626  104002                          ERROR   +2              ;MMU ERROR
8444                                                                 ;
8445 036630  020327  177700          14$:    CMP     R3,#177700              ;
8446 036634  001371                          BNE     13$                     ;
8447 036636  012703  172200                  MOV     #172200,R3              ;LOAD ALL PARS AND PDRS WITH ZEROES
8448 036642  012723  000000          15$:    MOV     #0,(R3)+                ;
8449 036646  020327  172400                  CMP     R3,#172400              ;
8450 036652  001373                          BNE     15$                     ;
8451 036654  012703  177600                  MOV     #177600,R3              ;
8452 036660  012723  000000          16$:    MOV     #0,(R3)+                ;
8453 036664  020327  177700                  CMP     R3,#177700              ;
8454 036670  001373                          BNE     16$                     ;
8455 036672  012703  172200                  MOV     #172200,R3              ;CHECK ALL PARS AND PDRS FOR ZEROES
8456 036676  022327  000000          17$:    CMP     (R3)+,#0                ;
8457 036702  001401                          BEQ     18$                     ;
8458 036704  104002                          ERROR   +2              ;MMU ERROR
```

MEMORY MANAGEMENT TESTS

```
                                                        ;
036706  020327  172400          18$:    CMP     R3,#172400              ;
036712  001371                          BNE     17$                     ;
036714  012703  177600                  MOV     #177600,R3              ;
036720  022327  000000          19$:    CMP     (R3)+,#0                ;
036724  001401                          BEQ     20$                     ;
036726  104002                          ERROR   +2              ;MMU ERROR
                                                        ;
036730  020327  177700          20$:    CMP     R3,#177700              ;
036734  001371                          BNE     19$                     ;

036736                          TSMMU4:
                                ;       TEST FOR ADJACENT SHORTS IN PARS/PDRS
036736  005037  177572                  CLR     @#177572                ;MMU OFF
036742  005067  144062                  CLR     FLAG                    ;CLEAR MMU ABORT FLAG
036746  012700  172200                  MOV     #172200,R0              ;LOAD SPDRS WITH ALTERNATING PATTERN
036752  012720  052404          1$:     MOV     #52404,(R0)+            ;
036756  012720  125012                  MOV     #125012,(R0)+           ;
036762  020027  172240                  CMP     R0,#172240              ;
036766  001371                          BNE     1$                      ;
036770  012720  125252          2$:     MOV     #125252,(R0)+           ;LOAD SPARS WITH ALTERNATING PATTERN
036774  012720  052525                  MOV     #52525,(R0)+            ;
037000  020027  172300                  CMP     R0,#172300              ;
037004  001371                          BNE     2$                      ;
037006  012720  052404          3$:     MOV     #52404,(R0)+            ;LOAD KPDRS WITH ALTERNATING PATTERN
037012  012720  125012                  MOV     #125012,(R0)+           ;
037016  020027  172340                  CMP     R0,#172340              ;
037022  001371                          BNE     3$                      ;
037024  012720  125252          4$:     MOV     #125252,(R0)+           ;LOAD KPARS WITH ALTERNATING PATTERN
037030  012720  052525                  MOV     #52525,(R0)+            ;
037034  020027  172400                  CMP     R0,#172400              ;
037040  001371                          BNE     4$                      ;
037042  012700  177600                  MOV     #177600,R0              ;LOAD UPDRS WITH ALTERNATING PATTERN
037046  012720  052404          5$:     MOV     #52404,(R0)+            ;
037052  012720  125012                  MOV     #125012,(R0)+           ;
037056  020027  177640                  CMP     R0,#177640              ;
037062  001371                          BNE     5$                      ;
037064  012720  125252          6$:     MOV     #125252,(R0)+           ;LOAD UPARS WITH ALTERNATING PATTERN
037070  012720  052525                  MOV     #52525,(R0)+            ;
037074  020027  177700                  CMP     R0,#177700              ;
037100  001371                          BNE     6$                      ;
                                ;
037102  012703  172200                  MOV     #172200,R3              ;CHECK SPDRS
037106  022327  052404          7$:     CMP     (R3)+,#52404            ;
037112  001401                          BEQ     8$                      ;
037114  104002                          ERROR   +2              ;MMU ERROR
                                                        ;
037116  022327  125012          8$:     CMP     (R3)+,#125012           ;
037122  001401                          BEQ     9$                      ;
037124  104002                          ERROR   +2              ;MMU ERROR
                                                        ;
037126  020327  172240          9$:     CMP     R3,#172240              ;
037132  001365                          BNE     7$                      ;
037134  022327  125252          10$:    CMP     (R3)+,#125252           ;CHECK SPARS
037140  001401                          BEQ     11$                     ;
037142  104002                          ERROR   +2              ;MMU ERROR
                                                        ;
```

MEMORY MANAGEMENT TESTS

```
8518 037144  022327  052525          11$:   CMP     (R3)+,#52525            ;
8519 037150  001401                        BEQ     12$                     ;
8520 037152  104002                        ERROR   +2              ;MMU ERROR
8521                                                               ;
8522 037154  020327  172300          12$:   CMP     R3,#172300              ;
8523 037160  001365                        BNE     10$                     ;
8524 037162  022327  052404          13$:   CMP     (R3)+,#52404            ;CHECK KPDRS
8525 037166  001401                        BEQ     14$                     ;
8526 037170  104002                        ERROR   +2              ;MMU ERROR
8527                                                               ;
8528 037172  022327  125012          14$:   CMP     (R3)+,#125012           ;
8529 037176  001401                        BEQ     15$                     ;
8530 037200  104002                        ERROR   +2              ;MMU ERROR
8531                                                               ;
8532 037202  020327  172340          15$:   CMP     R3,#172340              ;
8533 037206  001365                        BNE     13$                     ;
8534 037210  022327  125252          16$:   CMP     (R3)+,#125252           ;CHECK KPARS
8535 037214  001401                        BEQ     17$                     ;
8536 037216  104002                        ERROR   +2              ;MMU ERROR
8537                                                               ;
8538 037220  022327  052525          17$:   CMP     (R3)+,#52525            ;
8539 037224  001401                        BEQ     18$                     ;
8540 037226  104002                        ERROR   +2              ;MMU ERROR
8541                                                               ;
8542 037230  020327  172400-         18$:   CMP     R3,#172400              ;
8543 037234  001365                        BNE     16$                     ;
8544 037236  012703  177600                MOV     #177600,R3              ;CHECK UPDRS
8545 037242  022327  052404          19$:   CMP     (R3)+,#52404            ;
8546 037246  001401                        BEQ     20$                     ;
8547 037250  104002                        ERROR   +2              ;MMU ERROR
8548                                                               ;
8549 037252  022327  125012          20$:   CMP     (R3)+,#125012           ;
8550 037256  001401                        BEQ     21$                     ;
8551 037260  104002                        ERROR   +2              ;MMU ERROR
8552                                                               ;
8553 037262  020327  177640          21$:   CMP     R3,#177640              ;
8554 037266  001365                        BNE     19$                     ;
8555 037270  022327  125252          22$:   CMP     (R3)+,#125252           ;CHECK UPARS
8556 037274  001401                        BEQ     23$                     ;
8557 037276  104002                        ERROR   +2              ;MMU ERROR
8558                                                               ;
8559 037300  022327  052525          23$:   CMP     (R3)+,#52525            ;
8560 037304  001401                        BEQ     24$                     ;
8561 037306  104002                        ERROR   +2              ;MMU ERROR
8562                                                               ;
8563 037310  020327  177700          24$:   CMP     R3,#177700              ;
8564 037314  001365                        BNE     22$                     ;
8565                                ;
8566                                ;REVERSE ALTERNATING PATTERN
8567                                ;
8568 037316  012700  172200                MOV     #172200,R0              ;LOAD SPDRS WITH REVERSE PATTERN
8569 037322  012720  125012          25$:   MOV     #125012,(R0)+           ;
8570 037326  012720  052404                MOV     #52404,(R0)+            ;
8571 037332  020027  172240                CMP     R0,#172240              ;
8572 037336  001371                        BNE     25$                     ;
8573 037340  012720  052525          26$:   MOV     #52525,(R0)+            ;LOAD SPARS WITH REVERSE PATTERN
8574 037344  012720  125252                MOV     #125252,(R0)+           ;
```

MEMORY MANAGEMENT TESTS

```
8575 037350  020027  172300          CMP     R0,#172300              ;
8576 037354  001371                  BNE     26$                     ;
8577 037356  012720  125012    27$:   MOV     #125012,(R0)+           ;LOAD KPDRS WITH REVERSE PATTERN
8578 037362  012720  052404          MOV     #52404,(R0)+            ;
8579 037366  020027  172340          CMP     R0,#172340              ;
8580 037372  001371                  BNE     27$                     ;
8581 037374  012720  052525    28$:   MOV     #52525,(R0)+            ;LOAD KPARS WITH REVERSE PATTERN
8582 037400  012720  125252          MOV     #125252,(R0)+           ;
8583 037404  020027  172400          CMP     R0,#172400              ;
8584 037410  001371                  BNE     28$                     ;
8585 037412  012700  177600          MOV     #177600,R0              ;LOAD UPDRS WITH REVERSE PATTERN
8586 037416  012720  125012    29$:   MOV     #125012,(R0)+           ;
8587 037422  012720  052404          MOV     #52404,(R0)+            ;
8588 037426  020027  177640          CMP     R0,#177640              ;
8589 037432  001371                  BNE     29$                     ;
8590 037434  012720  052525    30$:   MOV     #52525,(R0)+            ;LOAD UPARS WITH REVERSE PATTERN
8591 037440  012720  125252          MOV     #125252,(R0)+           ;
8592 037444  020027  177700          CMP     R0,#177700              ;
8593 037450  001371                  BNE     30$                     ;
8594                               ;
8595 037452  012703  172200          MOV     #172200,R3              ;CHECK SPDRS
8596 037456  022327  125012    31$:   CMP     (R3)+,#125012           ;
8597 037462  001401                  BEQ     32$                     ;
8598 037464  104002                  ERROR   +2              ;MMU ERROR
8599                                                        ;
8600 037466  022327  052404    32$:   CMP     (R3)+,#52404            ;
8601 037472  001401                  BEQ     33$                     ;
8602 037474  104002                  ERROR   +2              ;MMU ERROR
8603                                                        ;
8604 037476  020327  172240    33$:   CMP     R3,#172240              ;
8605 037502  001365                  BNE     31$                     ;
8606 037504  022327  052525    34$:   CMP     (R3)+,#52525            ;CHECK SPARS
8607 037510  001401                  BEQ     35$                     ;
8608 037512  104002                  ERROR   +2              ;MMU ERROR
8609                                                        ;
8610 037514  022327  125252    35$:   CMP     (R3)+,#125252           ;
8611 037520  001401                  BEQ     36$                     ;
8612 037522  104002                  ERROR   +2              ;MMU ERROR
8613                                                        ;
8614 037524  020327  172300    36$:   CMP     R3,#172300              ;
8615 037530  001365                  BNE     34$                     ;
8616 037532  022327  125012    37$:   CMP     (R3)+,#125012           ;CHECK KPDRS
8617 037536  001401                  BEQ     38$                     ;
8618 037540  104002                  ERROR   +2              ;MMU ERROR
8619                                                        ;
8620 037542  022327  052404    38$:   CMP     (R3)+,#52404            ;
8621 037546  001401                  BEQ     39$                     ;
8622 037550  104002                  ERROR   +2              ;MMU ERROR
8623                                                        ;
8624 037552  020327  172340    39$:   CMP     R3,#172340              ;
8625 037556  001365                  BNE     37$                     ;
8626 037560  022327  052525    40$:   CMP     (R3)+,#52525            ;CHECK KPARS
8627 037564  001401                  BEQ     41$                     ;
8628 037566  104002                  ERROR   +2              ;MMU ERROR
8629                                                        ;
8630 037570  022327  125252    41$:   CMP     (R3)+,#125252           ;
8631 037574  001401                  BEQ     42$                     ;
```

MEMORY MANAGEMENT TESTS

```
8632 037576  104002                        ERROR   +2              ;MMU ERROR
8633                                                                ;
8634 037600  020327  172400          42$:   CMP     R3,#172400              ;
8635 037604  001365                         BNE     40$                     ;
8636 037606  012703  177600                 MOV     #177600,R3              ;CHECK UPDRS
8637 037612  022327  125012          43$:   CMP     (R3)+,#125012           ;
8638 037616  001401                         BEQ     44$                     ;
8639 037620  104002                         ERROR   +2              ;MMU ERROR
8640                                                                ;
8641 037622  022327  052404          44$:   CMP     (R3)+,#52404            ;
8642 037626  001401                         BEQ     45$                     ;
8643 037630  104002                         ERROR   +2              ;MMU ERROR
8644                                                                ;
8645 037632  020327  177640          45$:   CMP     R3,#177640              ;
8646 037636  001365                         BNE     43$                     ;
8647 037640  022327  052525          46$:   CMP     (R3)+,#52525            ;CHECK UPARS
8648 037644  001401                         BEQ     47$                     ;
8649 037646  104002                         ERROR   +2              ;MMU ERROR
8650                                                                ;
8651 037650  022327  125252          47$:   CMP     (R3)+,#125252           ;
8652 037654  001401                         BEQ     48$                     ;
8653 037656  104002                         ERROR   +2              ;MMU ERROR
8654                                                                ;
8655 037660  020327  177700          48$:   CMP     R3,#177700              ;
8656 037664  001365                         BNE     46$                     ;
8657
8660 037666                          TSMMU5:
8661                                 ;      TEST MMR0 ABORT BITS
8662 037666  012737  160000  177572         MOV     #160000,@#177572        ;LOAD MMR0<15:13>=111
8663 037674  005067  143130                 CLR     FLAG                    ;CLEAR MMU ABORT FLAG
8664 037700  013700  177572                 MOV     @#SR0,R0                ;SAVE SR0 IN R0
8665 037704  042700  000176                 BIC     #176,R0                 ;CLEAR UNDEFINED BITS FROM SR0
8666 037710  020027  160000                 CMP     R0,#160000              ;CHECK MMR0
8667 037714  001401                         BEQ     1$                      ;
8668 037716  104002                         ERROR   +2              ;MMU ERROR
8669                                                                ;
8670 037720  005037  177572          1$:    CLR     @#177572                ;LOAD MMR0=0
8671 037724  013700  177572                 MOV     @#SR0,R0                ;SAVE SR0 IN R0
8672 037730  042700  000176                 BIC     #176,R0                 ;CLEAR UNDEFINED BITS FROM SR0
8673 037734  020027  000000                 CMP     R0,#0                   ;CHECK MMR0
8674 037740  001401                         BEQ     2$                      ;
8675 037742  104002                         ERROR   +2              ;MMU ERROR
8676                                                                ;
8677 037744  012737  120000  177572  2$:    MOV     #120000,@#177572        ;LOAD MMR0<15:13>=101
8678 037752  013700  177572                 MOV     @#SR0,R0                ;SAVE SR0 IN R0
8679 037756  042700  000176                 BIC     #176,R0                 ;CLEAR UNDEFINED BITS FROM SR0.
8680 037762  020027  120000                 CMP     R0,#120000              ;CHECK MMR0
8681 037766  001401                         BEQ     3$                      ;
8682 037770  104002                         ERROR   +2              ;MMU ERROR
8683                                                                ;
8684 037772  012737  040000  177572  3$:    MOV     #40000,@#177572         ;LOAD MMR0<15:13>=010
8685 040000  013700  177572                 MOV     @#SR0,R0                ;SAVE SR0 IN R0
8686 040004  042700  000176                 BIC     #176,R0                 ;CLEAR UNDEFINED BITS FROM SR0.
8687 040010  020027  040000                 CMP     R0,#40000               ;CHECK MMR0
8688 040014  001401                         BEQ     4$                      ;
8689 040016  104002                         ERROR   +2              ;MMU ERROR
8690 040020                          4$:
```

MEMORY MANAGEMENT TESTS

```
 8693 040020                                TSMMU6:
 8694                                       ;       TEST MMR3 BITS 5-0
 8695 040020  005037  177572                        CLR     @#177572                ;MMU OFF
 8696 040024  005067  143000                        CLR     FLAG                    ;CLEAR MMU ABORT FLAG
 8697 040030  012737  000077  172516                MOV     #77,@#172516            ;LOAD MMR3<5:0>=77
 8698 040036  023727  172516  000077                CMP     @#172516,#77            ;CHECK MMR3
 8699 040044  001401                                BEQ     1$                      ;
 8700 040046  104002                                ERROR   .2              ;MMU ERROR
 8701 040050  005037  172516                1$:     CLR     @#172516                ;LOAD MMR3<5:0>=0
 8702 040054  023727  172516  000000                CMP     @#172516,#0             ;CHECK MMR3
 8703 040062  001401                                BEQ     2$                      ;
 8704 040064  104002                                ERROR   .2              ;MMU ERROR
 8705 040066  012737  000052  172516  2$:     MOV     #52,@#172516            ;LOAD MMR3<5:0>=52
 8706 040074  023727  172516  000052                CMP     @#172516,#52            ;CHECK MMR3
 8707 040102  001401                                BEQ     3$                      ;
 8708 040104  104002                                ERROR   .2              ;MMU ERROR
 8709 040106  012737  000025  172516  3$:     MOV     #25,@#172516            ;LOAD MMR3<5:0>=25
 8710 040114  023727  172516  000025                CMP     @#172516,#25            ;CHECK MMR3
 8711 040122  001401                                BEQ     4$                      ;
 8712 040124  104002                                ERROR   .2              ;MMU ERROR
 8713 040126                                4$:
 8716 040126                                TSMM6A:
 8717                                       ;       TEST MFPI (MOVE FROM PREVIOUS INST SPACE)
 8718 040126  005037  177572                        CLR     @#177572                ;MMU OFF
 8719 040132  005037  003030                        CLR     @#FLAG                  ;CLEAR MMU ABORT FLAG
 8720 040136  012737  140000  177776                MOV     #140000,@#177776        ;POINT TO USER SPACE
 8721 040144  012706  001000                        MOV     #STBOT,SP               ;INIT THE USER STACK POINTER
 8722 040150  010637  003040                        MOV     R6,@#SAVUSE             ;SAVE USER SP
 8723 040154  012737  040000  177776                MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
 8724 040162  012706  001000                        MOV     #STBOT,SP               ;INIT THE SUPERVISOR STACK POINTER
 8725 040166  010637  003036                        MOV     R6,@#SAVSUP             ;SAVE SUPERVISOR SP
 8726 040172  012737  030000  177776                MOV     #30000,@#177776         ;SETUP PSW
 8727 040200  004767  077346                        JSR     PC,MMU                  ;INIT MMU
 8728 040204  012737  000027  172516                MOV     #27,@#172516            ;SETUP MMR3
 8729 040212  013746  000244                        MOV     @#244,-(SP)             ;SAVE DATA AT TEST LOCATION
 8730 040216  012746  177777                        MOV     #177777,-(SP)           ;PUT KNOWN DATA ON TOP OF STACK
 8731 040222  012737  135072  000244                MOV     #135072,@#244           ;SETUP DATA AT TEST LOCATION
 8732 040230  012767  077400  137362                MOV     #77400,UDPDR0           ;SETUP UDPDR0 TO ABORT
 8733 040236  012703  000244                        MOV     #244,R3                 ;SETUP POINTER TO TEST LOCATION
 8734 040242  005237  177572                        INC     @#177572                ;TURN MMU ON
 8735 040246  006523                                MFPI    (R3)+                   ; TEST INSTRUCTION
 8736 040250  022737  030010  177776                CMP     #30010,@#177776         ;IS PSW CORRECT
 8737 040256  001401                                BEQ     1$                      ;YES GO ON
 8738 040260  104002                                ERROR   .2              ;MMU ERROR
 8739                                               ;NO GO TO ERROR
 8740 040262  005037  177572                1$:     CLR     @#177572                ;TURN MMU OFF
 8741 040266  012737  140000  177776                MOV     #140000,@#177776        ;POINT TO USER SPACE
 8742 040274  020637  003040                        CMP     R6,@#SAVUSE             ;IS USER SP CORRECT
 8743 040300  001401                                BEQ     100$                    ;YES GO ON
 8744 040302  104002                                ERROR   .2              ;MMU ERROR
 8745                                               ;NO GO TO ERROR
 8746 040304  012737  040000  177776  100$:   MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
 8747 040312  020637  003036                        CMP     R6,@#SAVSUP             ;IS SUPERVISOR SP CORRECT
 8748 040316  001401                                BEQ     200$                    ;YES GO ON
 8749 040320  104002                                ERROR   .2              ;MMU ERROR
 8750                                               ;NO GO TO ERROR
 8751 040322  023727  000244  135072  200$:   CMP     @#244,#135072           ;IS TEST DATA OK
```

MEMORY MANAGEMENT TESTS

```
8752 040330  001401                        BEQ     2$                      ;YES GO ON
8753 040332  104002                        ERROR   .+2                 ;MMU ERROR
8754                                                                   ;NO GO TO ERROR
8755 040334  020327  000246          2$:   CMP     R3,#246                 ;IS R3 CORRECT
8756 040340  001401                        BEQ     3$                      ;YES GO ON
8757 040342  104002                        ERROR   .+2                 ;MMU ERROR
8758                                                                   ;NO GO TO ERROR
8759 040344  005037  177776          3$:   CLR     @#177776                ;SET PSW TO KERNEL MODE
8760 040350  022627  135072                CMP     (SP)+,#135072           ;IS KERNEL STACK CORRECT
8761 040354  001401                        BEQ     4$                      ;YES GO ON
8762 040356  104002                        ERROR   .+2                 ;MMU ERROR
8763                                                                   ;NO GO TO ERROR
8764 040360  021627  177777          4$:   CMP     (SP),#177777            ;IS STACK CORRECT
8765 040364  001401                        BEQ     5$                      ;YES GO ON
8766 040366  104002                        ERROR   .+2                 ;MMU ERROR
8767                                                                   ;NO GO TO ERROR
8768 040370  012737  030017  177776  5$:   MOV     #30017,@#177776         ;SETUP PSW
8769 040376  012737  173621  000244        MOV     #173621,@#244           ;SETUP TEST LOCATION
8770 040404  012701  000244                MOV     #244,R1                 ;SETUP R1
8771 040410  005237  177572                INC     @#177572                ;TURN MMU ON
8772 040414  006511                        MFPI    (R1)                    ;TEST INSTRUCTION
8773 040416  022737  030011  177776        CMP     #30011,@#177776         ;IS PSW CORRECT
8774 040424  001401                        BEQ     300$                    ;YES GO ON
8775 040426  104002                        ERROR   .+2                 ;MMU ERROR
8776                                                                   ;NO GO TO ERROR
8777 040430  005037  177572          300$: CLR     @#177572                ;TURN MMU OFF
8778 040434  023727  000244  173621        CMP     @#244,#173621           ;IS TEST LOCATION CORRECT
8779 040442  001401                        BEQ     301$                    ;YES GO ON
8780 040444  104002                        ERROR   .+2                 ;MMU ERROR
8781                                                                   ;NO GO TO ERROR
8782 040446  020127  000244          301$: CMP     R1,#244                 ;IS R1 CORRECT
8783 040452  001401                        BEQ     302$                    ;YES GO ON
8784 040454  104002                        ERROR   .+2                 ;MMU ERROR
8785                                                                   ;NO GO TO ERROR
8786 040456  005037  177776          302$: CLR     @#177776                ;SET PSW TO KERNEL MODE
8787 040462  022627  173621                CMP     (SP)+,#173621           ;IS STACK CORRECT
8788 040466  001401                        BEQ     303$                    ;YES GO ON
8789 040470  104002                        ERROR   .+2                 ;MMU ERROR
8790                                                                   ;NO GO TO ERROR
8791 040472  021627  177777          303$: CMP     (SP),#177777            ;IS STACK CORRECT
8792 040476  001401                        BEQ     304$                    ;YES GO ON
8793 040500  104002                        ERROR   .+2                 ;MMU ERROR
8794                                                                   ;NO GO TO ERROR
8795 040502  005003                304$:   CLR     R3                      ;SETUP SOURCE FOR NEXT TEST
8796 040504  005237  177572                INC     @#177572                ;TURN MMU ON
8797 040510  006503                        MFPI    R3                      ; TEST INSTRUCTION
8798 040512  022737  000004  177776        CMP     #4,@#177776             ;IS PSW CORRECT
8799 040520  001401                        BEQ     6$                      ;YES GO ON
8800 040522  104002                        ERROR   .+2                 ;MMU ERROR
8801                                                                   ;NO GO TO ERROR
8802 040524  005037  177572          6$:   CLR     @#177572                ;TURN MMU OFF
8803 040530  020327  000000                CMP     R3,#0                   ;IS R3 CORRECT
8804 040534  001401                        BEQ     7$                      ;YES GO ON
8805 040536  104002                        ERROR   .+2                 ;MMU ERROR
8806                                                                   ;NO GO TO ERROR
8807 040540  022627  000000          7$:   CMP     (SP)+,#0                ;IS STACK CORRECT
8808 040544  001401                        BEQ     8$                      ;YES GO ON
```

MEMORY MANAGEMENT TESTS

```
8809 040546  104002                          ERROR   ;2              ;MMU ERROR
8810                                                                 ;NO GO TO ERROR
8811 040550  022627  177777          8$:     CMP     (SP)+,#177777           ;IS STACK CORRECT
8812 040554  001401                          BEQ     9$              ;YES GO ON
8813 040556  104002                          ERROR   ;2              ;MMU ERROR
8814                                                                 ;NO GO TO ERROR
8815 040560  012637  000244          9$:     MOV     (SP)+,@#244             ;RESTORE TEST LOCATION
8816
8817                                         ;
8820 040564                                  TSMM6B:
8821                                         ;       TEST MFPD (MOVE FROM PREVIOUS DATA SPACE)
8822 040564  005037  177572                  CLR     @#177572                ;MMU OFF
8823 040570  005037  003030                  CLR     @#FLAG                  ;CLEAR MMU ABORT FLAG
8824 040574  012737  140000  177776          MOV     #140000,@#177776        ;POINT TO USER SPACE
8825 040602  010637  003040                  MOV     R6,@#SAVUSE             ;SAVE USER SP
8826 040606  012737  040000  177776          MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
8827 040614  010637  003036                  MOV     R6,@#SAVSUP             ;SAVE SUPERVISOR SP
8828 040620  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
8829 040626  004767  076720                  JSR     PC,MMU                  ;INIT MMU
8830 040632  012737  000027  172516          MOV     #27,@#172516            ;SETUP MMR3
8831 040640  013746  000244                  MOV     @#244,-(SP)             ;SAVE DATA AT TEST LOCATION
8832 040644  012746  177777                  MOV     #177777,-(SP)           ;PUT KNOWN DATA ON TOP OF STACK
8833 040650  012737  157002  000244          MOV     #157002,@#244           ;SETUP DATA AT TEST LOCATION
8834 040656  012767  077400  136714          MOV     #77400,UIPDR0           ;SETUP UIPDR0 TO ABORT
8835 040664  012703  000244                  MOV     #244,R3                 ;SETUP POINTER TO TEST LOCATION
8836 040670  005237  177572                  INC     @#177572                ;TURN MMU ON
8837 040674  106523                          MFPD    (R3)+                   ; TEST INSTRUCTION
8838 040676  022737  030010  177776          CMP     #30010,@#177776         ;IS PSW CORRECT
8839 040704  001401                          BEQ     1$              ;YES GO ON
8840 040706  104002                          ERROR   ;2              ;MMU ERROR
8841                                                                 ;NO GO TO ERROR
8842 040710  005037  177572          1$:     CLR     @#177572                ;TURN MMU OFF
8843 040714  012737  140000  177776          MOV     #140000,@#177776        ;POINT TO USER SPACE
8844 040722  020637  003040                  CMP     R6,@#SAVUSE             ;IS USER SP CORRECT
8845 040726  001401                          BEQ     100$            ;YES GO ON
8846 040730  104002                          ERROR   ;2              ;MMU ERROR
8847                                                                 ;NO GO TO ERROR
8848 040732  012737  040000  177776  100$:   MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
8849 040740  020637  003036                  CMP     R6,@#SAVSUP             ;IS SUPERVISOR SP CORRECT
8850 040744  001401                          BEQ     200$            ;YES GO ON
8851 040746  104002                          ERROR   ;2              ;MMU ERROR
8852                                                                 ;NO GO TO ERROR
8853 040750  023727  000244  157002  200$:   CMP     @#244,#157002           ;IS TEST DATA OK
8854 040756  001401                          BEQ     2$              ;YES GO ON
8855 040760  104002                          ERROR   ;2              ;MMU ERROR
8856                                                                 ;NO GO TO ERROR
8857 040762  020327  000246          2$:     CMP     R3,#246                 ;IS R3 CORRECT
8858 040766  001401                          BEQ     3$              ;YES GO ON
8859 040770  104002                          ERROR   ;2              ;MMU ERROR
8860                                                                 ;NO GO TO ERROR
8861 040772  005037  177776          3$:     CLR     @#177776                ;SET PSW TO KERNEL MODE
8862 040776  022627  157002                  CMP     (SP)+,#157002           ;IS KERNEL STACK CORRECT
8863 041002  001401                          BEQ     4$              ;YES GO ON
8864 041004  104002                          ERROR   ;2              ;MMU ERROR
8865                                                                 ;NO GO TO ERROR
8866 041006  021627  177777          4$:     CMP     (SP),#177777            ;IS STACK CORRECT
8867 041012  001401                          BEQ     5$              ;YES GO ON
```

MEMORY MANAGEMENT TESTS

```
8868 041014  104002                          ERROR   ;2              ;MMU ERROR
8869                                                                 ;NO GO TO ERROR
8870 041016  012737  030017  177776  5$:     MOV     #30017,@#177776         ;SETUP PSW
8871 041024  012737  103456  000244          MOV     #103456,@#244           ;SETUP TEST LOCATION
8872 041032  012701  000244                  MOV     #244,R1                 ;SETUP R1
8873 041036  005237  177572                  INC     @#177572                ;TURN MMU ON
8874 041042  106511                          MFPD    (R1)                    ;TEST INSTRUCTION
8875 041044  022737  030011  177776          CMP     #30011,@#177776         ;IS PSW CORRECT
8876 041052  001401                          BEQ     300$                    ;YES GO ON
8877 041054  104002                          ERROR   ;2              ;MMU ERROR
8878                                                                 ;NO GO TO ERROR
8879 041056  005037  177572          300$:   CLR     @#177572                ;TURN MMU OFF
8880 041062  023727  000244  103456          CMP     @#244,#103456           ;IS TEST LOCATION CORRECT
8881 041070  001401                          BEQ     301$                    ;YES GO ON
8882 041072  104002                          ERROR   ;2              ;MMU ERROR
8883                                                                 ;NO GO TO ERROR
8884 041074  020127  000244          301$:   CMP     R1,#244                 ;IS R1 CORRECT
8885 041100  001401                          BEQ     302$                    ;YES GO ON
8886 041102  104002                          ERROR   ;2              ;MMU ERROR
8887                                                                 ;NO GO TO ERROR
8888 041104  005037  177776          302$:   CLR     @#177776                ;SET PSW TO KERNEL MODE
8889 041110  022627  103456                  CMP     (SP)+,#103456           ;IS STACK CORRECT
8890 041114  001401                          BEQ     303$                    ;YES GO ON
8891 041116  104002                          ERROR   ;2              ;MMU ERROR
8892                                                                 ;NO GO TO ERROR
8893 041120  021627  177777          303$:   CMP     (SP),#177777            ;IS STACK CORRECT
8894 041124  001401                          BEQ     304$                    ;YES GO ON
8895 041126  104002                          ERROR   ;2              ;MMU ERROR
8896                                                                 ;NO GO TO ERROR
8897 041130  012737  030017  177776  304$:   MOV     #30017,@#177776         ;SETUP PSW
8898 041136  012737  113672  000244          MOV     #113672,@#244           ;SETUP TEST LOCATION
8899 041144  012701  000246                  MOV     #246,R1                 ;SETUP R1
8900 041150  005237  177572                  INC     @#177572                ;TURN MMU ON
8901 041154  106541                          MFPD    -(R1)                   ;TEST INSTRUCTION
8902 041156  022737  030011  177776          CMP     #30011,@#177776         ;IS PSW CORRECT
8903 041164  001401                          BEQ     400$                    ;YES GO ON
8904 041166  104002                          ERROR   ;2              ;MMU ERROR
8905                                                                 ;NO GO TO ERROR
8906 041170  005037  177572          400$:   CLR     @#177572                ;TURN MMU OFF
8907 041174  023727  000244  113672          CMP     @#244,#113672           ;IS TEST LOCATION CORRECT
8908 041202  001401                          BEQ     401$                    ;YES GO ON
8909 041204  104002                          ERROR   ;2              ;MMU ERROR
8910                                                                 ;NO GO TO ERROR
8911 041206  020127  000244          401$:   CMP     R1,#244                 ;IS R1 CORRECT
8912 041212  001401                          BEQ     402$                    ;YES GO ON
8913 041214  104002                          ERROR   ;2              ;MMU ERROR
8914                                                                 ;NO GO TO ERROR
8915 041216  005037  177776          402$:   CLR     @#177776                ;SET PSW TO KERNEL MODE
8916 041222  022627  113672                  CMP     (SP)+,#113672           ;IS STACK CORRECT
8917 041226  001401                          BEQ     403$                    ;YES GO ON
8918 041230  104002                          ERROR   ;2              ;MMU ERROR
8919                                                                 ;NO GO TO ERROR
8920 041232  021627  177777          403$:   CMP     (SP),#177777            ;IS STACK CORRECT
8921 041236  001401                          BEQ     404$                    ;YES GO ON
8922 041240  104002                          ERROR   ;2              ;MMU ERROR
8923                                                                 ;NO GO TO ERROR
8924 041242  005003                  404$:   CLR     R3                      ;SETUP SOURCE FOR NEXT TEST
```

MEMORY MANAGEMENT TESTS

```
 8925 041244  005237  177572              INC     @#177572                ;TURN MMU ON
 8926 041250  106503                      MFPD    R3                      ; TEST INSTRUCTION
 8927 041252  022737  000004  177776      CMP     #4,@#177776             ;IS PSW CORRECT
 8928 041260  001401                      BEQ     6$                      ;YES GO ON
 8929 041262  104002                      ERROR   +2              ;MMU ERROR
 8930                                                             ;NO GO TO ERROR
 8931 041264  005037  177572      6$:     CLR     @#177572                ;TURN MMU OFF
 8932 041270  020327  000000              CMP     R3,#0                   ;IS R3 CORRECT
 8933 041274  001401                      BEQ     7$                      ;YES GO ON
 8934 041276  104002                      ERROR   +2              ;MMU ERROR
 8935                                                             ;NO GO TO ERROR
 8936 041300  022627  000000      7$:     CMP     (SP)+,#0                ;IS STACK CORRECT
 8937 041304  001401                      BEQ     8$                      ;YES GO ON
 8938 041306  104002                      ERROR   +2              ;MMU ERROR
 8939                                                             ;NO GO TO ERROR
 8940 041310  022627  177777      8$:     CMP     (SP)+,#177777           ;IS STACK CORRECT
 8941 041314  001401                      BEQ     9$                      ;YES GO ON
 8942 041316  104002                      ERROR   +2              ;MMU ERROR
 8943                                                             ;NO GO TO ERROR
 8944 041320  012637  000244      9$:     MOV     (SP)+,@#244             ;RESTORE TEST LOCATION
 8945
 8946                             ;
 8949 041324                      TSMM6C:
 8950                             ;       TEST MTPI (MOVE TO PREVIOUS INSTRUCTION SPACE)
 8951 041324  005037  177572              CLR     @#177572                ;MMU OFF
 8952 041330  005037  003030              CLR     @#FLAG                  ;CLEAR MMU ABORT FLAG
 8953 041334  012737  140000  177776      MOV     #140000,@#177776        ;POINT TO USER SPACE
 8954 041342  010637  003040              MOV     R6,@#SAVUSE             ;SAVE USER SP
 8955 041346  012737  040000  177776      MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
 8956 041354  010637  003036              MOV     R6,@#SAVSUP             ;SAVE SUPERVISOR SP
 8957 041360  012737  030000  177776      MOV     #30000,@#177776         ;SETUP PSW
 8958 041366  004767  076160              JSR     PC,MMU                  ;INIT MMU
 8959 041372  012737  000027  172516      MOV     #27,@#172516            ;SETUP MMR3
 8960 041400  013746  000244              MOV     @#244,-(SP)             ;SAVE DATA AT TEST LOCATION
 8961 041404  012746  177777              MOV     #177777,-(SP)           ;PUT KNOWN DATA ON STACK
 8962 041410  012746  120413              MOV     #120413,-(SP)           ;PUT TEST DATA ON STACK
 8963 041414  012737  177777  000244      MOV     #177777,@#244           ;PUT KNOWN DATA AT TEST LOCATION
 8964 041422  012767  077400  136170      MOV     #77400,UDPDR0           ;SETUP UDPDR0 TO ABORT
 8965 041430  012703  000244              MOV     #244,R3                 ;SETUP POINTER TO TEST LOCATION
 8966 041434  005237  177572              INC     @#177572                ;TURN MMU ON
 8967 041440  006623                      MTPI    (R3)+                   ; TEST INSTRUCTION
 8968 041442  022737  030010  177776      CMP     #30010,@#177776         ;IS PSW CORRECT
 8969 041450  001401                      BEQ     1$                      ;YES GO ON
 8970 041452  104002                      ERROR   +2              ;MMU ERROR
 8971                                                             ;NO GO TO ERROR
 8972 041454  005037  177572      1$:     CLR     @#177572                ;TURN MMU OFF
 8973 041460  012737  140000  177776      MOV     #140000,@#177776        ;POINT TO USER SPACE
 8974 041466  020637  003040              CMP     R6,@#SAVUSE             ;IS USER SP CORRECT
 8975 041472  001401                      BEQ     100$                    ;YES GO ON
 8976 041474  104002                      ERROR   +2              ;MMU ERROR
 8977                                                             ;NO GO TO ERROR
 8978 041476  012737  040000  177776  100$:   MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
 8979 041504  020637  003036              CMP     R6,@#SAVSUP             ;IS SUPERVISOR SP CORRECT
 8980 041510  001401                      BEQ     200$                    ;YES GO ON
 8981 041512  104002                      ERROR   +2              ;MMU ERROR
 8982                                                             ;NO GO TO ERROR
 8983 041514  023727  000244  120413  200$:   CMP     @#244,#120413           ;IS TEST LOCATION CORRECT
```

MEMORY MANAGEMENT TESTS

```
 8984 041522  001401                        BEQ     2$                ;YES GO ON
 8985 041524  104002                        ERROR   .+2             ;MMU ERROR
 8986                                                               ;NO GO TO ERROR
 8987 041526  020327  000246          2$:   CMP     R3,#246           ;IS R3 CORRECT
 8988 041532  001401                        BEQ     3$                ;YES GO ON
 8989 041534  104002                        ERROR   .+2             ;MMU ERROR
 8990                                                               ;NO GO TO ERROR
 8991 041536  005037  177776          3$:   CLR     @#177776          ;SET PSW TO KERNEL MODE
 8992 041542  021627  177777                CMP     (SP),#177777      ;IS KERNEL STACK CORRECT
 8993 041546  001401                        BEQ     4$                ;YES GO ON
 8994 041550  104002                        ERROR   .+2             ;MMU ERROR
 8995                                                               ;NO GO TO ERROR
 8996 041552  012737  030017  177776  4$:   MOV     #30017,@#177776   ;SETUP PSW
 8997 041560  012746  145121                MOV     #145121,-(SP)     ;SETUP TEST DATA
 8998 041564  012701  000244                MOV     #244,R1           ;SETUP R1
 8999 041570  005237  177572                INC     @#177572          ;TURN MMU ON
 9000 041574  006611                        MTPI    (R1)              ;TEST INSTRUCTION
 9001 041576  022737  030011  177776        CMP     #30011,@#177776   ;IS PSW CORRECT
 9002 041604  001401                        BEQ     300$              ;YES GO ON
 9003 041606  104002                        ERROR   .+2             ;MMU ERROR
 9004                                                               ;NO GO TO ERROR
 9005 041610  005037  177572          300$: CLR     @#177572          ;TURN MMU OFF
 9006 041614  023727  000244  145121        CMP     @#244,#145121     ;IS TEST LOCATION CORRECT
 9007 041622  001401                        BEQ     301$              ;YES GO ON
 9008 041624  104002                        ERROR   .+2             ;MMU ERROR
 9009                                                               ;NO GO TO ERROR
 9010 041626  020127  000244          301$: CMP     R1,#244           ;IS R1 CORRECT
 9011 041632  001401                        BEQ     302$              ;YES GO ON
 9012 041634  104002                        ERROR   .+2             ;MMU ERROR
 9013                                                               ;NO GO TO ERROR
 9014 041636  005037  177776          302$: CLR     @#177776          ;SET PSW TO KERNEL MODE
 9015 041642  021627  177777                CMP     (SP),#177777      ;IS STACK CORRECT
 9016 041646  001401                        BEQ     304$              ;YES GO ON
 9017 041650  104002                        ERROR   .+2             ;MMU ERROR
 9018                                                               ;NO GO TO ERROR
 9019 041652  012737  030017  177776  304$: MOV     #30017,@#177776   ;SETUP PSW
 9020 041660  012746  122347                MOV     #122347,-(SP)     ;SETUP TEST DATA
 9021 041664  012701  000246                MOV     #246,R1           ;SETUP R1
 9022 041670  005237  177572                INC     @#177572          ;TURN MMU ON
 9023 041674  006641                        MTPI    -(R1)             ;TEST INSTRUCTION
 9024 041676  022737  030011  177776        CMP     #30011,@#177776   ;IS PSW CORRECT
 9025 041704  001401                        BEQ     400$              ;YES GO ON
 9026 041706  104002                        ERROR   .+2             ;MMU ERROR
 9027                                                               ;NO GO TO ERROR
 9028 041710  005037  177572          400$: CLR     @#177572          ;TURN MMU OFF
 9029 041714  023727  000244  122347        CMP     @#244,#122347     ;IS TEST LOCATION CORRECT
 9030 041722  001401                        BEQ     401$              ;YES GO ON
 9031 041724  104002                        ERROR   .+2             ;MMU ERROR
 9032                                                               ;NO GO TO ERROR
 9033 041726  020127  000244          401$: CMP     R1,#244           ;IS R1 CORRECT
 9034 041732  001401                        BEQ     402$              ;YES GO ON
 9035 041734  104002                        ERROR   .+2             ;MMU ERROR__
 9036                                                               ;NO GO TO ERROR
 9037 041736  005037  177776          402$: CLR     @#177776          ;SET PSW TO KERNEL MODE
 9038 041742  021627  177777                CMP     (SP),#177777      ;IS STACK CORRECT
 9039 041746  001401                        BEQ     404$              ;YES GO ON
 9040 041750  104002                        ERROR   .+2             ;MMU ERROR
```

MEMORY MANAGEMENT TESTS

```
                                                                ;NO GO TO ERROR
042752  005046                  404$:   CLR     -(SP)                   ;SETUP STACK FOR NEXT TEST
042754  005237  177572                  INC     @#177572                ;TURN MMU ON
042760  006603                          MTPI    R3                      ; TEST INSTRUCTION
042762  022737  000004  177776          CMP     #4,@#177776             ;IS PSW CORRECT
042770  001401                          BEQ     5$                      ;YES GO ON
042772  104002                          ERROR   +2              ;MMU ERROR
                                                                ;NO GO TO ERROR
042774  005037  177572          5$:     CLR     @#177572                ;TURN MMU OFF
042000  020327  000000                  CMP     R3,#0                   ;IS R3 CORRECT
042004  001401                          BEQ     6$                      ;YES GO ON
042006  104002                          ERROR   +2              ;MMU ERROR
                                                                ;NO GO TO ERROR
042010  022627  177777          6$:     CMP     (SP)+,#177777           ;IS STACK CORRECT
042014  001401                          BEQ     7$                      ;YES GO ON
042016  104002                          ERROR   -+2             ;MMU ERROR
                                                                ;NO GO TO ERROR
042020  012637  000244          7$:     MOV     (SP)+,@#244             ;RESTORE TEST LOCATION

                                ;
042024                          TSMM6D:
                                ;       TEST MTPD (MOVE TO PREVIOUS DATA SPACE)
042024  005037  177572                  CLR     @#177572                ;MMU OFF
042030  005037  003030                  CLR     @#FLAG                  ;CLEAR MMU ABORT FLAG
042034  012737  140000  177776          MOV     #140000,@#177776        ;POINT TO USER SPACE
042042  010637  003040                  MOV     R6,@#SAVUSE             ;SAVE USER SP
042046  012737  040000  177776          MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
042054  010637  003036                  MOV     R6,@#SAVSUP             ;SAVE SUPERVISOR SP
042060  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
042066  004767  075460                  JSR     PC,MMU                  ;INIT MMU
042072  012737  000027  172516          MOV     #27,@#172516            ;SETUP MMR3
042100  013746  000244                  MOV     @#244,-(SP)             ;SAVE DATA AT TEST LOCATION
042104  012746  177777                  MOV     #177777,-(SP)           ;PUT KNOWN DATA ON STACK
042110  012746  100004                  MOV     #100004,-(SP)           ;PUT TEST DATA ON STACK
042114  012737  177777  000244          MOV     #177777,@#244           ;PUT KNOWN DATA AT TEST LOCATION
042122  012767  077400  135450          MOV     #77400,UIPDR0           ;SETUP UIPDR0 TO ABORT
042130  012703  000244                  MOV     #244,R3                 ;SETUP POINTER TO TEST LOCATION
042134  005237  177572                  INC     @#177572                ;TURN MMU ON
042140  106623                          MTPD    (R3)+                   ; TEST INSTRUCTION
042142  022737  030010  177776          CMP     #30010,@#177776         ;IS PSW CORRECT
042150  001401                          BEQ     1$                      ;YES GO ON
042152  104002                          ERROR   +2              ;MMU ERROR
                                                                ;NO GO TO ERROR
042154  005037  177572          1$:     CLR     @#177572                ;TURN MMU OFF
042160  012737  140000  177776          MOV     #140000,@#177776        ;POINT TO USER SPACE
042166  020637  003040                  CMP     R6,@#SAVUSE             ;IS USER SP CORRECT
042172  001401                          BEQ     100$                    ;YES GO ON
042174  104002                          ERROR   +2              ;MMU ERROR
                                                                ;NO GO TO ERROR
042176  012737  040000  177776  100$:   MOV     #40000,@#177776         ;POINT TO SUPERVISOR SPACE
042204  020637  003036                  CMP     R6,@#SAVSUP             ;IS SUPERVISOR SP CORRECT
042210  001401                          BEQ     200$                    ;YES GO ON
042212  104002                          ERROR   +2              ;MMU ERROR
                                                                ;NO GO TO ERROR
042214  023727  000244  100004  200$:   CMP     @#244,#100004           ;IS TEST LOCATION CORRECT
042222  001401                          BEQ     2$                      ;YES GO ON
042224  104002                          ERROR   +2              ;MMU ERROR
```

MEMORY MANAGEMENT TESTS

```
9100
9101 042226  020327  000246          2$:     CMP     R3,#246                 ;NO GO TO ERROR
9102 042232  001401                          BEQ     3$                              ;IS R3 CORRECT
9103 042234  104002                          ERROR   +2                              ;YES GO ON
9104                                                                        ;MMU ERROR
9105 042236  005037  177776          3$:     CLR     @#177776                ;NO GO TO ERROR
9106 042242  021627  177777                  CMP     (SP),#177777                    ;SET PSW TO KERNEL MODE
9107 042246  001401                          BEQ     4$                              ;IS KERNEL STACK CORRECT
9108 042250  104002                          ERROR   +2                              ;YES GO ON
9109                                                                        ;MMU ERROR
9110 042252  012737  030017  177776  4$:     MOV     #30017,@#177776         ;NO GO TO ERROR
9111 042260  012746  100737                  MOV     #100737,-(SP)                   ;SETUP PSW
9112 042264  012701  000244                  MOV     #244,R1                         ;SETUP TEST DATA
9113 042270  005237  177572                  INC     @#177572                        ;SETUP R1
9114 042274  106611                          MTPD    (R1)                            ;TURN MMU ON
9115 042276  022737  030011  177776          CMP     #30011,@#177776                 ;TEST INSTRUCTION
9116 042304  001401                          BEQ     300$                            ;IS PSW CORRECT
9117 042306  104002                          ERROR   +2                              ;YES GO ON
9118                                                                        ;MMU ERROR
9119 042310  005037  177572          300$:   CLR     @#177572                ;NO GO TO ERROR
9120 042314  023727  000244  100737          CMP     @#244,#100737                   ;TURN MMU OFF
9121 042322  001401                          BEQ     301$                            ;IS TEST LOCATION CORRECT
9122 042324  104002                          ERROR   +2                              ;YES GO ON
9123                                                                        ;MMU ERROR
9124 042326  020127  000244          301$:   CMP     R1,#244                 ;NO GO TO ERROR
9125 042332  001401                          BEQ     302$                            ;IS R1 CORRECT
9126 042334  104002                          ERROR   +2                              ;YES GO ON
9127                                                                        ;MMU ERROR
9128 042336  005037  177776          302$:   CLR     @#177776                ;NO GO TO ERROR
9129 042342  021627  177777                  CMP     (SP),#177777                    ;SET PSW TO KERNEL MODE
9130 042346  001401                          BEQ     304$                            ;IS STACK CORRECT
9131 042350  104002                          ERROR   +2                              ;YES GO ON
9132                                                                        ;MMU ERROR
9133 042352  012737  030017  177776  304$:   MOV     #30017,@#177776         ;NO GO TO ERROR
9134 042360  012746  156711                  MOV     #156711,-(SP)                   ;SETUP PSW
9135 042364  012701  000246                  MOV     #246,R1                         ;SETUP TEST DATA
9136 042370  005237  177572                  INC     @#177572                        ;SETUP R1
9137 042374  106641                          MTPD    -(R1)                           ;TURN MMU ON
9138 042376  022737  030011  177776          CMP     #30011,@#177776                 ;TEST INSTRUCTION
9139 042404  001401                          BEQ     400$                            ;IS PSW CORRECT
9140 042406  104002                          ERROR   +2                              ;YES GO ON
9141                                                                        ;MMU ERROR
9142 042410  005037  177572          400$:   CLR     @#177572                ;NO GO TO ERROR
9143 042414  023727  000244  156711          CMP     @#244,#156711                   ;TURN MMU OFF
9144 042422  001401                          BEQ     401$                            ;IS TEST LOCATION CORRECT
9145 042424  104002                          ERROR   +2                              ;YES GO ON
9146                                                                        ;MMU ERROR
9147 042426  020127  000244          401$:   CMP     R1,#244                 ;NO GO TO ERROR
9148 042432  001401                          BEQ     402$                            ;IS R1 CORRECT
9149 042434  104002                          ERROR   +2                              ;YES GO ON
9150                                                                        ;MMU ERROR
9151 042436  005037  177776          402$:   CLR     @#177776                ;NO GO TO ERROR
9152 042442  021627  177777                  CMP     (SP),#177777                    ;SET PSW TO KERNEL MODE
9153 042446  001401                          BEQ     404$                            ;IS STACK CORRECT
9154 042450  104002                          ERROR   +2                              ;YES GO ON
9155                                                                        ;MMU ERROR
9156 042452  005046                  404$:   CLR     -(SP)                   ;NO GO TO ERROR
                                                                                    ;SETUP STACK FOR NEXT TEST
```

MEMORY MANAGEMENT TESTS

```
 9157 042454  005237  177572                INC     @#177572                ;TURN MMU ON
 9158 042460  106603                        MTPD    R3                      ; TEST INSTRUCTION
 9159 042462  022737  000004  177776        CMP     #4,@#177776             ;IS PSW CORRECT
 9160 042470  001401                        BEQ     5$                      ;YES GO ON
 9161 042472  104002                        ERROR   .+2                 ;MMU ERROR
 9162                                                                   ;NO GO TO ERROR
 9163 042474  005037  177572          5$:   CLR     @#177572                ;TURN MMU OFF
 9164 042500  020327  000000                CMP     R3,#0                   ;IS R3 CORRECT
 9165 042504  001401                        BEQ     6$                      ;YES GO ON
 9166 042506  104002                        ERROR   .+2                 ;MMU ERROR
 9167                                                                   ;NO GO TO ERROR
 9168 042510  022627  177777          6$:   CMP     (SP)+,#177777           ;IS STACK CORRECT
 9169 042514  001401                        BEQ     7$                      ;YES GO ON
 9170 042516  104002                        ERROR   .+2                 ;MMU ERROR
 9171                                                                   ;NO GO TO ERROR
 9172 042520  012637  000244          7$:   MOV     (SP)+,@#244             ;RESTORE TEST LOCATION
 9173
 9174                                       ;
 9177 042524                        TSMMU7:
 9178                                       ;       TEST NON-RESIDENT ABORT
 9179 042524  005037  177572                CLR     @#177572                ;MMU OFF
 9180 042530  005067  140274                CLR     FLAG                    ;CLEAR MMU ABORT FLAG
 9181 042534  013746  000214                MOV     @#214,-(SP)             ;SAVE DATA AT TEST LOCATIONS
 9182 042540  013746  000216                MOV     @#216,-(SP)             ;
 9183 042544  005067  140272                CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
 9184 042550  005067  140270                CLR     SAVMR1                  ;
 9185 042554  005067  140266                CLR     SAVMR2                  ;
 9186 042560  004767  074766                JSR     PC,MMU                  ;INIT MMU
 9187 042564  012737  030000  177776        MOV     #30000,@#177776         ;SETUP PSW
 9188 042572  012702  000200                MOV     #200,R2                 ;
 9189 042576  012737  077400  177600        MOV     #77400,@#177600         ;SETUP FOR AN ABORT
 9190 042604  004767  000164                JSR     PC,TS7                  ;CAUSE AN ABORT TO OCCUR AND
 9191                                                                   ;THEN CHECK IF ABORT FLAG REGISTERED
 9192                                                                   ;THIS EVENT AND CHECK IF STATUS REGS
 9193                                                                   ;CONTAINED EXPECTED VALUES.
 9194                                                                   ;IF NO ABORT OCCURRED THEN GO TO ERROR
 9195                                                                   ;OTHERWISE CONTINUE.
 9196 042610  012737  077404  177600        MOV     #77404,@#177600         ;SETUP FOR AN ABORT
 9197 042616  004767  000152                JSR     PC,TS7                  ;CAUSE AN ABORT TO OCCUR AND
 9198                                                                   ;THEN CHECK IF ABORT FLAG REGISTERED
 9199                                                                   ;THIS EVENT AND CHECK IF STATUS REGS
 9200                                                                   ;CONTAINED EXPECTED VALUES.
 9201                                                                   ;IF NO ABORT OCCURRED THEN GO TO ERROR
 9202                                                                   ;OTHERWISE CONTINUE.
 9203 042622  012701  000220                MOV     #220,R1                 ;
 9204 042626  004767  074720                JSR     PC,MMU                  ;INIT MMU
 9205 042632  005003                        CLR     R3                      ;SETUP MMR1 EXPECTED DATA
 9206 042634  012767  000001  140166        MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
 9207 042642  012737  000001  177572        MOV     #1,@#177572             ;TURN MMU ON
 9208 042650  012737  100000  177776        MOV     #100000,@#177776        ;SETUP PSW FOR AN ABORT (ILLEGAL MODE)
 9209 042656  012241                        MOV     (R2)+,-(R1)             ;CAUSE AN ABORT
 9210 042660  004767  000220                JSR     PC,TSM7                 ;CHECK IF AN ABORT OCCURRED BY
 9211                                                                   ;CHECKING ABORT FLAG AND STATUS REGS
 9212                                                                   ;IF NO ABORT OCCURRED THEN GO TO ERROR
 9213                                                                   ;OTHERWISE CONTINUE.
 9214 042664  005067  140152                CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
 9215 042670  005067  140150                CLR     SAVMR1                  ;
```

MEMORY MANAGEMENT TESTS

```
 9216 042674  005067  140146                CLR     SAVMR2                  ;
 9217 042700  012703  000022                MOV     #22,R3                  ;SETUP MMR1 EXPECTED DATA
 9218 042704  012767  000001  140116        MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
 9219 042712  012737  000001  177572        MOV     #1,@#177572             ;TURN MMU ON
 9220 042720  012737  020000  177776        MOV     #20000,@#177776         ;SETUP PSW FOR AN ABORT (ILLEGAL MODE)
 9221 042726  006522                        MFPI    (R2)+                   ;CAUSE AN ABORT
 9222 042730  004767  000150                JSR     PC,TSM7                 ;CHECK IF AN ABORT OCCURRED BY
 9223                                                                       ;CHECKING ABORT FLAG AND STATUS REGS
 9224                                                                       ;IF NO ABORT OCCURRED THEN GO TO ERROR
 9225                                                                       ;OTHERWISE CONTINUE.
 9226 042734  012737  030000  177776        MOV     #30000,@#177776         ;SETUP PSW
 9227 042742  012737  077400  177600        MOV     #77400,@#177600         ;SETUP FOR AN ABORT
 9228 042750  005037  177572                CLR     @#177572                ;MMU OFF
 9229 042754  006522                        MFPI    (R2)+                   ;TRY TO CAUSE AN ABORT
 9230 042756  012603                        MOV     (SP)+,R3                ;POP THE STACK
 9231 042760  012637  000216                MOV     (SP)+,@#216             ;RESTORE DATA AT TEST LOCATIONS
 9232 042764  012637  000214                MOV     (SP)+,@#214             ;
 9233
 9234 042770  000167  000154                JMP     TS7FIN
 9235                                ;
 9236                                ;ROUTINE TO CAUSE AND CHECK  NONRESIDENT ABORTS
 9237                                ;
 9238 042774  012767  000001  140026  TS7:   MOV     #1,FLAG                 ;SETUP FOR AN ABORT
 9239 043002  012737  000001  177572        MOV     #1,@#177572             ;TURN MMU ON
 9240 043010  010701                        MOV     R7,R1                   ;SAVE PC
 9241 043012  006522                        MFPI    (R2)+                   ;CAUSE AN ABORT
 9242 043014  022767  000000  140006        CMP     #0,FLAG                 ;DID AN ABORT OCCUR
 9243 043022  001401                        BEQ     OK7                     ;IF YES GO ON
 9244 043024  104002                        ERROR   +2              ;MMU ERROR
 9245                                                                ;IF NO GO TO ERROR
 9246 043026  105067  140010        OK7:    CLRB    SAVMR0                  ;SETUP EXPECTED DATA
 9247 043032  022767  100000  140002        CMP     #100000,SAVMR0          ; TEST MMR0 FOR EXPECTED VALUE
 9248 043040  001401                        BEQ     OKA7                    ;IF OK THEN CONTINUE
 9249 043042  104002                        ERROR   +2              ;MMU ERROR
 9250                                                                ;NOT OK THEN GO TO ERROR
 9251 043044  026727  137774  000022  OKA7:   CMP     SAVMR1,#22              ; TEST MMR1 FOR EXPECTED VALUE
 9252 043052  001401                        BEQ     OKAY7                   ;IF OK THEN CONTINUE
 9253 043054  104002                        ERROR   +2              ;MMU ERROR
 9254                                                                ;NOT OK THEN GO TO ERROR
 9255 043056  026701  137764        OKAY7:  CMP     SAVMR2,R1               ; TEST MMR2 FOR EXPECTED VALUE
 9256 043062  001401                        BEQ     OKAY7A                  ;IF OK THEN CONTINUE
 9257 043064  104002                        ERROR   +2              ;MMU ERROR
 9258                                                                ;NOT OK THEN GO TO ERROR
 9259 043066  005067  137750        OKAY7A: CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
 9260 043072  005067  137746                CLR     SAVMR1                  ;
 9261 043076  005067  137744                CLR     SAVMR2                  ;
 9262 043102  000207                        RTS     PC                      ;RETURN
 9263                                ;
 9264                                ;ROUTINE TO CHECK IF A NONRESIDENT ABORT OCCURRED
 9265                                ;
 9266 043104  022767  000000  137716  TSM7:   CMP     #0,FLAG                 ;DID AN ABORT OCCUR
 9267 043112  001401                        BEQ     TSMA                    ;IF YES GO ON
 9268 043114  104002                        ERROR   +2              ;MMU ERROR
 9269                                                                ;IF NO THEN GO TO ERROR
 9270 043116  042737  040377  003042  TSMA:   BIC     #40377,@#SAVMR0         ;SETUP EXPECTED DATA
 9271 043124  022767  100000  137710        CMP     #100000,SAVMR0          ; TEST MMR0 FOR EXPECTED VALUE
 9272 043132  001401                        BEQ     TSMB                    ;IF OK THEN CONTINUE
```

MEMORY MANAGEMENT TESTS

```
9273 043134  104002                        ERROR   +2              ;MMU ERROR
9274                                                                ;IF NO THEN GO TO ERROR
9275 043136  020367  137702          TSMB:  CMP     R3,SAVMR1               ; TEST MMR1 FOR EXPECTED VALUE
9276 043142  001401                 BEQ     TSMC                    ;IF OK THEN CONTINUE
9277 043144  104002                 ERROR   +2              ;MMU ERROR
9278                                                                ;IF NOT OK THEN GO TO ERROR
9279 043146  000207          TSMC:  RTS     PC                      ;RETURN
9280                                ;
9281 043150  000240          TS7FIN: NOP
9284 043152                  TSMMU8:
9285                                ;       TEST READ ONLY ABORTS
9286 043152  005037  177572            CLR     @#177572                ;MMU OFF
9287 043156  005067  137646            CLR     FLAG                    ;CLEAR MMU ABORT FLAG
9288 043162  013746  000244            MOV     @#244,-(SP)             ;SAVE DATA AT TEST LOCATIONS
9289 043166  013746  000246            MOV     @#246,-(SP)             ;
9290 043172  005067  137644            CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
9291 043176  005067  137642            CLR     SAVMR1                  ;
9292 043202  005067  137640            CLR     SAVMR2                  ;
9293 043206  004767  074340            JSR     PC,MMU                  ;INIT MMU
9294 043212  012737  030000  177776      MOV     #30000,@#177776         ;SETUP PSW
9295 043220  012702  000244            MOV     #244,R2                 ;
9296 043224  012737  077402  177600      MOV     #77402,@#177600         ;SETUP FOR AN ABORT
9297 043232  012746  000246            MOV     #246,-(SP)              ;PUSH DATA ONTO THE STACK
9298 043236  012767  000001  137564      MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
9299 043244  012737  000001  177572      MOV     #1,@#177572             ;TURN MMU ON
9300 043252  010701                    MOV     R7,R1                   ;SAVE PC
9301 043254  006622                    MTPI    (R2)+                   ;CAUSE ABORT
9302 043256  022767  000000  137544      CMP     #0,FLAG                 ;DID ABORT OCCUR
9303 043264  001401                    BEQ     1$                      ;IF YES THEN GO ON
9304 043266  104002                    ERROR   +2              ;MMU ERROR
9305                                                                ;IF NO THEN GO TO ERROR
9306 043270  105067  137546      1$:   CLRB    SAVMR0                  ;SETUP EXPECTED DATA
9307 043274  022767  020000  137540      CMP     #20000,SAVMR0           ; TEST MMR0 FOR EXPECTED VALUE
9308 043302  001401                    BEQ     2$                      ;IF OK THEN CONTINUE
9309 043304  104002                    ERROR   +2              ;MMU ERROR
9310                                                                ;OTHERWISE GO TO ERROR
9311 043306  022767  011026  137530  2$:   CMP     #11026,SAVMR1           ; TEST MMR1 FOR EXPECTED VALUE
9312 043314  001401                    BEQ     3$                      ;IF OK THEN CONTINUE
9313 043316  104002                    ERROR   +2              ;MMU ERROR
9314                                                                ;OTHERWISE GO TO ERROR
9315 043320  020167  137522      3$:   CMP     R1,SAVMR2               ; TEST MMR2 FOR EXPECTED VALUE
9316 043324  001401                    BEQ     4$                      ;IF OK THEN CONTINUE
9317 043326  104002                    ERROR   +2              ;MMU ERROR
9318                                                                ;OTHERWISE GO TO ERROR
9319 043330  012737  030000  177776  4$:   MOV     #30000,@#177776         ;SETUP PSW
9320 043336  012746  000002            MOV     #2,-(SP)                ;PUSH DATA ONTO STACK
9321 043342  006622                    MTPI    (R2)+                   ;TRY TO CAUSE ABORT
9322 043344  012637  000246            MOV     (SP)+,@#246             ;RESTORE DATA AT TEST LOCATIONS
9323 043350  012637  000244            MOV     (SP)+,@#244             ;
9324
9327 043354                  TSMMU9:
9328                                ;       TEST PAGE LENGTH ERROR ABORTS
9329 043354  005037  177572            CLR     @#177572                ;MMU OFF
9330 043360  005067  137444            CLR     FLAG                    ;CLEAR MMU ABORT FLAG
9331 043364  005067  137452            CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
9332 043370  005067  137450            CLR     SAVMR1                  ;
9333 043374  005067  137446            CLR     SAVMR2                  ;
```

MEMORY MANAGEMENT TESTS

```
9334 043400  012737  030000  177776          MOV     #30000,@#177776         ;SETUP PSW
9335 043406  004767  074140                  JSR     PC,MMU                  ;INIT MMU
9336 043412  012703  043674                  MOV     #PLF0,R3                ;LET R3, R1, AND R2 POINT TO THE
9337 043416  012701  043746                  MOV     #BN0,R1                 ;UPWARD EXPANSION TABLES
9338 043422  012702  044016                  MOV     #ABORT0,R2              ;
9339 043426  012737  000026  172516          MOV     #26,@#172516            ;DISABLE USER DATA SPACE
9340 043434  004767  000050                  JSR     PC,TSM9                 ;TURN MMU ON
9341                                                                         ;DO RELOCATIONS FOR THE DIFFERENT
9342                                                                         ;VALUES OF THE PAGE LENGTH FIELD AND
9343                                                                         ;BLOCK NUMBER.  IF AN ABORT OCCURS
9344                                                                         ;CHECK TO SEE IF IT WAS SUPPOSED TO,
9345                                                                         ;AND IF YES CHECK ABORT FLAG AND
9346                                                                         ;STATUS REGISTERS.
9347 043440  012703  044066                  MOV     #PLF1,R3                ;LET R3, R1, AND R2 POINT TO THE
9348 043444  012701  044136                  MOV     #BN1,R1                 ;DOWNWARD EXPANSION TABLES
9349 043450  012702  044204                  MOV     #ABORT7,R2              ;
9350 043454  004767  000030                  JSR     PC,TSM9                 ;TURN MMU ON
9351                                                                         ;DO RELOCATIONS FOR THE DIFFERENT
9352                                                                         ;VALUES OF THE PAGE LENGTH FIELD AND
9353                                                                         ;BLOCK NUMBER.  IF AN ABORT OCCURS
9354                                                                         ;CHECK TO SEE IF IT WAS SUPPOSED TO,
9355                                                                         ;AND IF YES CHECK ABORT FLAG AND
9356                                                                         ;STATUS REGISTERS.
9357 043460  005037  177572                  CLR     @#177572                ;MMU OFF
9358 043464  012703  044076                  MOV     #PLF1+10,R3             ;POINT TO A VALUE WHICH SHOULD CAUSE
9359 043470  012701  044146                  MOV     #BN1+10,R1              ;AN ABORT IF MMU IS ON.
9360 043474  011337  177600                  MOV     (R3),@#177600           ;SETUP UIPDR0
9361 043500  006521                          MFPI    (R1)+                   ;DO A RELOCATION
9362 043502  012605                          MOV     (SP)+,R5                ;POP THE STACK
9363
9364 043504  000167  000542                  JMP     TS9FIN
9365                                 ;
9366                                 ;ROUTINE TO CAUSE AND CHECK PAGE LENGTH ERROR ABORTS
9367                                 ;
9368 043510  012337  177600          TSM9:   MOV     (R3)+,@#177600          ;SETUP UIPDR0
9369 043514  010100                          MOV     R1,R0                   ;SAVE A COPY OF R1
9370 043516  012767  000001  137304          MOV     #1,FLAG                 ;SETUP FOR AN ABORT
9371 043524  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
9372 043532  010704                          MOV     R7,R4                   ;SAVE PC
9373 043534  006530                          MFPI    @(R0)+                  ;DO A RELOCATION OPERATION
9374 043536  021227  000000                  CMP     (R2),#0                 ;WAS AN ABORT SUPPOSED TO OCCUR
9375 043542  001007                          BNE     2$                      ;IF YES GO TO 2$
9376 043544  012605                          MOV     (SP)+,R5                ;POP THE STACK
9377 043546  022767  000001  137254          CMP     #1,FLAG                 ;DID AN ABORT OCCUR
9378 043554  001401                          BEQ     1$                      ;NO GO ON
9379 043556  104002                          ERROR   +2              ;MMU ERROR
9380                                                                 ;YES GO TO ERROR
9381 043560  000425          1$:     BR      6$                      ;
9382 043562  022767  000000  137240  2$:     CMP     #0,FLAG                 ;DID AN ABORT OCCUR
9383 043570  001401                          BEQ     3$                      ;YES GO ON
9384 043572  104002                          ERROR   +2              ;MMU ERROR
9385                                                                 ;NO GO TO ERROR
9386 043574  105067  137242          3$:     CLRB    SAVMR0                  ;SETUP EXPECTED DATA
9387 043600  022767  040000  137234          CMP     #40000,SAVMR0           ; TEST MMR0 FOR EXPECTED VALUE
9388 043606  001401                          BEQ     4$                      ;IF OK THEN CONTINUE
9389 043610  104002                          ERROR   +2              ;MMU ERROR
9390                                                                 ;NOT OK THEN GO TO ERROR
```

MEMORY MANAGEMENT TESTS

```
 9391 043612  022767  000020  137224  4$:     CMP     #20,SAVMR1              ; TEST MMR1 FOR EXPECTED VALUE
 9392 043620  001401                         BEQ     5$                      ;IF OK THEN CONTINUE
 9393 043622  104002                         ERROR   .+2                  ;MMU ERROR
 9394                                                                     ;NOT OK THEN GO TO ERROR
 9395 043624  020467  137216          5$:     CMP     R4,SAVMR2               ; TEST MMR2 FOR EXPECTED VALUE
 9396 043630  001401                         BEQ     6$                      ;IF OK THEN CONTINUE
 9397 043632  104002                         ERROR   .+2                  ;MMU ERROR
 9398                                                                     ;NOT OK THEN GO TO ERROR
 9399 043634  005067  137170          6$:     CLR     FLAG                    ;CLEAR MMU ABORT FLAG
 9400 043640  005067  137176                  CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
 9401 043644  005067  137174                  CLR     SAVMR1                  ;
 9402 043650  005067  137172                  CLR     SAVMR2                  ;
 9403 043654  005201                         INC     R1                      ;POINT TO NEXT ENTRY
 9404 043656  005201                         INC     R1                      ;
 9405 043660  005202                         INC     R2                      ;
 9406 043662  005202                         INC     R2                      ;
 9407 043664  021327  000777                CMP     (R3),#777               ;HAVE ALL ENTRIES BEEN TRIED
 9408 043670  001307                         BNE     TSM9                    ;NO REPEAT
 9409 043672  000207                         RTS     PC                      ;YES RETURN
 9410                                 ;
 9411                                 ;UPWARD EXPANSION TABLES
 9412                                 ;
 9413 043674  070006                 PLF0:   .WORD   70006
 9414 043676  070006                         .WORD   70006
 9415 043700  070006                         .WORD   70006
 9416 043702  013406                         .WORD   13406
 9417 043704  020006                         .WORD   20006
 9418 043706  004006                         .WORD   04006
 9419 043710  040006                         .WORD   40006
 9420 043712  070006                         .WORD   70006
 9421 043714  024006                         .WORD   24006
 9422 043716  004006                         .WORD   04006
 9423 043720  014006                         .WORD   14006
 9424 043722  012006                         .WORD   12006
 9425 043724  002006                         .WORD   02006
 9426 043726  001406                         .WORD   01406
 9427 043730  004006                         .WORD   04006
 9428 043732  002006                         .WORD   02006
 9429 043734  000406                         .WORD   00406
 9430 043736  007406                         .WORD   07406
 9431 043740  001006                         .WORD   01006
 9432 043742  003406                         .WORD   03406
 9433 043744  000777                         .WORD   777
 9434 043746  013000                 BN0:    .WORD   013000
 9435 043750  016000                         .WORD   016000
 9436 043752  017000                         .WORD   017000
 9437 043754  002700                         .WORD   002700
 9438 043756  014000                         .WORD   014000
 9439 043760  002000                         .WORD   002000
 9440 043762  004000                         .WORD   004000
 9441 043764  007000                         .WORD   007000
 9442 043766  002000                         .WORD   002000
 9443 043770  000700                         .WORD   000700
 9444 043772  004000                         .WORD   004000
 9445 043774  001000                         .WORD   001000
 9446 043776  000300                         .WORD   000300
 9447 044000  000400                         .WORD   000400
```

MEMORY MANAGEMENT TESTS

```
 9448 044002  001400                        .WORD   001400
 9449 044004  000600                        .WORD   000600
 9450 044006  000200                        .WORD   000200
 9451 044010  001700                        .WORD   001700
 9452 044012  000300                        .WORD   000300
 9453 044014  000700                        .WORD   000700
 9454 044016  000000                ABORTO: .WORD   0
 9455 044020  000000                        .WORD   0
 9456 044022  000001                        .WORD   1
 9457 044024  000000                        .WORD   0
 9458 044026  000001                        .WORD   1
 9459 044030  000001                        .WORD   1
 9460 044032  000000                        .WORD   0
 9461 044034  000000                        .WORD   0
 9462 044036  000000                        .WORD   0
 9463 044040  000000                        .WORD   0
 9464 044042  000001                        .WORD   1
 9465 044044  000000                        .WORD   0
 9466 044046  000000                        .WORD   0
 9467 044050  000001                        .WORD   1
 9468 044052  000001                        .WORD   1
 9469 044054  000001                        .WORD   1
 9470 044056  000001                        .WORD   1
 9471 044060  000000                        .WORD   0
 9472 044062  000001                        .WORD   1
 9473 044064  000000                        .WORD   0
 9474                               ;
 9475                               ;DOWNWARD EXPANSION TABLES
 9476                               ;
 9477 044066  000416                PLF1:   .WORD   00416
 9478 044070  020016                        .WORD   20016
 9479 044072  024016                        .WORD   24016
 9480 044074  034016                        .WORD   34016
 9481 044076  074016                        .WORD   74016
 9482 044100  040016                        .WORD   40016
 9483 044102  020016                        .WORD   20016
 9484 044104  000016                        .WORD   00016
 9485 044106  030016                        .WORD   30016
 9486 044110  010016                        .WORD   10016
 9487 044112  014016                        .WORD   14016
 9488 044114  004016                        .WORD   04016
 9489 044116  002016                        .WORD   02016
 9490 044120  000416                        .WORD   00416
 9491 044122  000016                        .WORD   00016
 9492 044124  003416                        .WORD   03416
 9493 044126  001016                        .WORD   01016
 9494 044130  001416                        .WORD   01416
 9495 044132  000416                        .WORD   00416
 9496 044134  000777                        .WORD   777
 9497 044136  000100                BN1:    .WORD   000100
 9498 044140  010000                        .WORD   010000
 9499 044142  006000                        .WORD   006000
 9500 044144  016000                        .WORD   016000
 9501 044146  016000                        .WORD   016000
 9502 044150  004000                        .WORD   004000
 9503 044152  000000                        .WORD   000000
 9504 044154  000000                        .WORD   000000
```

MEMORY MANAGEMENT TESTS

```
 9505 044156  004000                          .WORD   004000
 9506 044160  004000                          .WORD   004000
 9507 044162  004000                          .WORD   004000
 9508 044164  000000                          .WORD   000000
 9509 044166  000300                          .WORD   000300
 9510 044170  000000                          .WORD   000000
 9511 044172  000400                          .WORD   000400
 9512 044174  001000                          .WORD   001000
 9513 044176  000100                          .WORD   000100
 9514 044200  000400                          .WORD   000400
 9515 044202  000200                          .WORD   000200
 9516 044204  000000                ABORT7:   .WORD   0
 9517 044206  000000                          .WORD   0
 9518 044210  000000                          .WORD   0
 9519 044212  000000                          .WORD   0
 9520 044214  000001                          .WORD   1
 9521 044216  000001                          .WORD   1
 9522 044220  000001                          .WORD   1
 9523 044222  000000                          .WORD   0
 9524 044224  000001                          .WORD   1
 9525 044226  000000                          .WORD   0
 9526 044230  000000                          .WORD   0
 9527 044232  000001                          .WORD   1
 9528 044234  000001                          .WORD   1
 9529 044236  000001                          .WORD   1
 9530 044240  000000                          .WORD   0
 9531 044242  000000                          .WORD   0
 9532 044244  000001                          .WORD   1
 9533 044246  000000                          .WORD   0
 9534 044250  000000                          .WORD   0
 9535                               ;
 9536 044252  000240                TS9FIN:   NOP
 9539 044254                        TSMM10:
 9540                               ;         FUNCTIONAL TEST OF BITS <6:1> OF MMR0
 9541 044254  005037  177572                  CLR     @#177572                ;MMU OFF
 9542 044260  005067  136544                  CLR     FLAG                    ;CLEAR MMU ABORT FLAG
 9543 044264  005067  136552                  CLR     SAVMR0                  ;CLEAR STATUS REGS SAVE AREAS
 9544 044270  005067  136550                  CLR     SAVMR1                  ;
 9545 044274  005067  136546                  CLR     SAVMR2                  ;
 9546 044300  004767  073246                  JSR     PC,MMU                  ;INIT MMU
 9547 044304  005037  177776                  CLR     @#177776                ;INIT PSW: PREVIOUS MODE = KERNAL
 9548 044310  012702  020200                  MOV     #20200,R2               ;
 9549 044314  012737  077400  172302          MOV     #77400,@#172302         ;SETUP KIPDR1 TO ABORT
 9550 044322  012767  000001  136500          MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
 9551 044330  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
 9552 044336  010701                          MOV     R7,R1                   ;SAVE PC
 9553 044340  006522                          MFPI    (R2)+                   ;DO A RELOCATION VIA KIPAR1
 9554 044342  012704  100003                  MOV     #100003,R4              ;SETUP EXPECTED DATA
 9555 044346  004767  000202                  JSR     PC,TS10                 ;CHECK IF AN ABORT OCCURRED AND
 9556                                                                         ;IF YES CHECK BITS <6:1> OF MMR0.
 9557 044352  012737  030000  177776          MOV     #30000,@#177776         ;INIT PSW: PREVIOUS MODE = USER
 9558 044360  004767  073166                  JSR     PC,MMU                  ;INIT MMU
 9559 044364  012737  077400  177636          MOV     #77400,@#177636         ;SETUP UDPDR7 TO ABORT
 9560 044372  012702  160000                  MOV     #160000,R2              ;
 9561 044376  012767  000001  136424          MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
 9562 044404  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
 9563 044412  010701                          MOV     R7,R1                   ;SAVE PC
```

MEMORY MANAGEMENT TESTS

```
 9564 044414  106522                        MFPD    (R2)+                   ;DO A RELOCATION VIA UDPAR7
 9565 044416  012704  100177                MOV     #100177,R4              ;SETUP EXPECTED DATA
 9566 044422  004767  000126                JSR     PC,TS10                 ;CHECK IF AN ABORT OCCURRED AND
 9567                                                                        ;IF YES CHECK BITS <6:1> OF MMR0.
 9568 044426  012737  010000  177776        MOV     #10000,@#177776         ;INIT PSW: PREVIOUS MODE= SUPERVISOR
 9569 044434  004767  073112                JSR     PC,MMU                  ;INIT MMU
 9570 044440  012737  077400  172212        MOV     #77400,@#172212         ;SETUP SIPDR5 TO ABORT
 9571 044446  012702  120000                MOV     #120000,R2              ;ACCESS PAGE 05
 9572 044452  012767  000001  136350        MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
 9573 044460  012737  000001  177572        MOV     #1,@#177572             ;TURN MMU ON
 9574 044466  010701                        MOV     R7,R1                   ;SAVE PC
 9575 044470  006522                        MFPI    (R2)+                   ;DO A RELOCATION VIA SIPAR5
 9576 044472  012704  100053                MOV     #100053,R4              ;SETUP EXPECTED DATA:ABORT, PAGE 05
 9577 044476  004767  000052                JSR     PC,TS10                 ;CHECK IF AN ABORT OCCURRED AND
 9578                                                                        ;IF YES CHECK BITS <6:1> OF MMR0.
 9579                                ;
 9580                                ;TEST THAT ILLEGAL MODE CAUSES MMU ABORT
 9581                                ;
 9582 044502  012737  020000  177776        MOV     #20000,@#177776         ;INIT PSW:SET ILLEGAL PREVIOUS MODE
 9583 044510  004767  073036                JSR     PC,MMU                  ;INIT MMU
 9584 044514  012702  040000                MOV     #40000,R2               ;SET UP ACCESS TO PAGE 2
 9585 044520  012767  000001  136302        MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
 9586 044526  012737  000001  177572        MOV     #1,@#177572             ;TURN MMU ON
 9587 044534  010701                        MOV     R7,R1                   ;SAVE PC
 9588 044536  106522                        MFPD    (R2)+                   ;DO A RELOCATION
 9589 044540  012704  100105                MOV     #100105,R4              ;SETUP EXPECTED DATA:ABORT, ILLEGAL
 9590                                                                        ; PROCESSOR MODE, PAGE 02
 9591 044544  004767  000004                JSR     PC,TS10                 ;CHECK IF AN ABORT OCCURRED AND
 9592                                                                        ;IF YES CHECK BITS <6:1> OF MMR0.
 9593
 9594 044550  000167  000062                JMP     T10FIN
 9595                                ;
 9596                                ;ROUTINE TO CHECK IF A MMU ABORT OCCURRED AND IF STATUS REG MMR0
 9597                                ;CONTAINS EXPECTED DATA
 9598                                ;
 9599 044554  022767  000000  136246  TS10:   CMP     #0,FLAG                         ;DID AN ABORT OCCUR
 9600 044562  001401                        BEQ     1$                              ;YES GO ON
 9601 044564  104002                        ERROR   .+2                     ;MMU ERROR
 9602                                                                       ;NO GO TO ERROR
 9603 044566  020467  136250          1$:   CMP     R4,SAVMR0                       ; TEST MMR0 FOR EXPECTED DATA
 9604 044572  001401                        BEQ     2$                              ;OK GO ON
 9605 044574  104002                        ERROR   .+2                     ;MMU ERROR
 9606                                                                       ;NO GO TO ERROR
 9607 044576  022767  000022  136240  2$:   CMP     #22,SAVMR1                      ; TEST MMR1 FOR EXPECTED DATA
 9608 044604  001401                        BEQ     3$                              ;OK GO ON
 9609 044606  104002                        ERROR   .+2                     ;MMU ERROR
 9610                                                                       ;NO GO TO ERROR
 9611 044610  020167  136232          3$:   CMP     R1,SAVMR2                       ; TEST MMR2 FOR EXPECTED DATA
 9612 044614  001401                        BEQ     4$                              ;OK GO ON
 9613 044616  104002                        ERROR   .+2                     ;MMU ERROR
 9614                                                                       ;NO GO TO ERROR
 9615 044620  005067  136216          4$:   CLR     SAVMR0                          ;CLEAR MMU STATUS REGS SAVE AREAS
 9616 044624  005067  136214                CLR     SAVMR1                          ;
 9617 044630  005067  136212                CLR     SAVMR2                          ;
 9618 044634  000207                        RTS     PC                              ;RETURN
 9619                                ;
 9620 044636                        T10FIN:
```

MEMORY MANAGEMENT TESTS

```
 9623 044636                                TSMM11:
 9624                                       ;       TEST DATA SPACE BITS MMR3
 9625 044636  005037  177572                        CLR     @#177572                ;MMU OFF
 9626 044642  005067  136162                        CLR     FLAG                    ;CLEAR MMU ABORT FLAG
 9627 044646  012737  030000  177776                MOV     #30000,@#177776         ;SETUP PSW
 9628 044654  012701  000026                        MOV     #26,R1                  ;SETUP FIRST MMR3 VALUE
 9629 044660  012703  177610                        MOV     #177610,R3              ;POINT TO UIPDR4
 9630 044664  012704  000021                        MOV     #21,R4                  ;SETUP SECOND MMR3 VALUE
 9631 044670  004767  000060                        JSR     PC,TS11                 ; TEST ENABLE USER DATA SPACE BIT
 9632 044674  012737  000000  177776                MOV     #0,@#177776             ;SETUP PSW
 9633 044702  012701  000023                        MOV     #23,R1                  ;SETUP FIRST MMR3 VALUE
 9634 044706  012703  172310                        MOV     #172310,R3              ;POINT TO KIPDR4
 9635 044712  012704  000024                        MOV     #24,R4                  ;SETUP SECOND MMR3 VALUE
 9636 044716  004767  000032                        JSR     PC,TS11                 ; TEST ENABLE KERNEL DATA SPACE BIT
 9637 044722  012737  010000  177776                MOV     #10000,@#177776         ;SETUP PSW
 9638 044730  012701  000025                        MOV     #25,R1                  ;SETUP FIRST MMR3 VALUE
 9639 044734  012703  172210                        MOV     #172210,R3              ;POINT TO SIPDR4
 9640 044740  012704  000022                        MOV     #22,R4                  ;SETUP SECOND MMR3 VALUE
 9641 044744  004767  000004                        JSR     PC,TS11                 ; TEST ENABLE SUPERVISOR DATA SPACE BIT
 9642
 9643 044750  000167  000120                        JMP     T11FIN
 9644                                       ;
 9645                                       ;ROUTINE TO TEST ENABLE DATA SPACE BITS OF MMR3
 9646                                       ;
 9647 044754  004767  072572                TS11:   JSR     PC,MMU                  ;INIT MMU
 9648 044760  010137  172516                        MOV     R1,@#172516             ;DISABLE DATA SPACE OF MODE UNDER TEST
 9649 044764  012713  077400                        MOV     #77400,(R3)             ;SETUP IPDR TO ABORT
 9650 044770  012702  100000                        MOV     #100000,R2              ;
 9651 044774  012767  000001  136026                MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
 9652 045002  012737  000001  177572                MOV     #1,@#177572             ;MMU ON
 9653 045010  106522                                MFPD    (R2)+                   ;DO A RELOCATION
 9654 045012  022767  000000  136010                CMP     #0,FLAG                 ;DID AN ABORT OCCUR
 9655 045020  001401                                BEQ     1$                      ;YES GO ON
 9656 045022  104002                                ERROR   +2              ;MMU ERROR
 9657                                               ;NO GO TO ERROR
 9658 045024  010437  172516                1$:     MOV     R4,@#172516             ;ENABLE DATA SPACE OF MODE UNDER TEST
 9659 045030  012702  100000                        MOV     #100000,R2              ;
 9660 045034  012767  000001  135766                MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
 9661 045042  012737  000001  177572                MOV     #1,@#177572             ;MMU ON
 9662 045050  106522                                MFPD    (R2)+                   ;DO A RELOCATION
 9663 045052  005726                                TST     (SP)+                   ;POP THE STACK
 9664 045054  022767  000001  135746                CMP     #1,FLAG                 ;DID AN ABORT OCCUR
 9665 045062  001401                                BEQ     2$                      ;NO GO ON
 9666 045064  104002                                ERROR   +2              ;MMU ERROR
 9667                                               ;YES GO TO ERROR
 9668 045066  005067  135736                2$:     CLR     FLAG                    ;CLEAR MMU ABORT FLAG
 9669 045072  000207                                RTS     PC                      ;RETURN
 9670                                       ;
 9671 045074                                T11FIN:
 9674 045074                                TSMM12:
 9675                                       ;       MMR1 FUNCTIONAL TEST
 9676 045074  005037  177572                        CLR     @#177572                ;MMU OFF
 9677 045100  005067  135724                        CLR     FLAG                    ;CLEAR MMU ABORT FLAG
 9678 045104  005067  135734                        CLR     SAVMR1                  ;CLEAR STATUS REG SAVE AREA
 9679 045110  004767  072436                        JSR     PC,MMU                  ;INIT MMU
 9680 045114  012737  030000  177776                MOV     #30000,@#177776         ;INIT PSW
 9681 045122  012704  100200                        MOV     #100200,R4              ;SETUP TEST LOCATIONS
```

MEMORY MANAGEMENT TESTS

```
    9682 045126  010401                    MOV     R4,R1                   ;
    9683 045130  012705  100101            MOV     #100101,R5              ;
    9684 045134  010502                    MOV     R5,R2                   ;
    9685 045136  012737  000020  172516    MOV     #20,@#172516            ;INIT MMR3
    9686 045144  012737  077402  172310    MOV     #77402,@#172310         ;SETUP KIPDR4 TO ABORT
    9687 045152  012703  006414            MOV     #6414,R3                ;SETUP EXPECTED DATA FOR MMR1
    9688 045156  012767  000001  135644    MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
    9689 045164  012737  000001  177572    MOV     #1,@#177572             ;TURN MMU ON
Z   9690 045172  010767  135614            MOV     R7,SLOC00               ;SAVE PC
    9691 045176  112425                    MOVB    (R4)+,(R5)+             ;DO A RELOCATION
    9692 045200  004767  000206            JSR     PC,TS12                 ;CHECK IF AN ABORT OCCURRED AND IF
    9693                                                                    ;YES IF MMR1 EQUALS EXPECTED DATA
    9694 045204  012703  175011            MOV     #175011,R3              ;SETUP EXPECTED DATA FOR MMR1
    9695 045210  012767  000001  135612    MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
    9696 045216  012737  000001  177572    MOV     #1,@#177572             ;TURN MMU ON
Z   9697 045224  010767  135562            MOV     R7,SLOC00               ;SAVE PC
    9698 045230  112142                    MOVB    (R1)+,-(R2)             ;DO A RELOCATION
    9699 045232  004767  000154            JSR     PC,TS12                 ;CHECK IF AN ABORT OCCURRED AND IF
    9700                                                                    ;YES IF MMR1 EQUALS EXPECTED DATA
    9701 045236  012703  006771            MOV     #6771,R3                ;SETUP EXPECTED DATA FOR MMR1
    9702 045242  012767  000001  135560    MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
    9703 045250  012737  000001  177572    MOV     #1,@#177572             ;TURN MMU ON
Z   9704 045256  010767  135530            MOV     R7,SLOC00               ;SAVE PC
    9705 045262  114125                    MOVB    -(R1),(R5)+             ;DO A RELOCATION
    9706 045264  004767  000122            JSR     PC,TS12                 ;CHECK IF AN ABORT OCCURRED AND IF
    9707                                                                    ;YES IF MMR1 EQUALS EXPECTED DATA
    9708 045270  012703  006411            MOV     #6411,R3                ;SETUP EXPECTED DATA FOR MMR1
    9709 045274  012767  000001  135526    MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
    9710 045302  012737  000001  177572    MOV     #1,@#177572             ;TURN MMU ON
Z   9711 045310  010767  135476            MOV     R7,SLOC00               ;SAVE PC
    9712 045314  112125                    MOVB    (R1)+,(R5)+             ;DO A RELOCATION
    9713 045316  004767  000070            JSR     PC,TS12                 ;CHECK IF AN ABORT OCCURRED AND IF
    9714                                                                    ;YES IF MMR1 EQUALS EXPECTED DATA
    9715 045322  012703  171025            MOV     #171025,R3              ;SETUP EXPECTED DATA FOR MMR1
    9716 045326  012767  000001  135474    MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
    9717 045334  012737  000001  177572    MOV     #1,@#177572             ;TURN MMU ON
Z   9718 045342  010767  135444            MOV     R7,SLOC00               ;SAVE PC
    9719 045346  012542                    MOV     (R5)+,-(R2)             ;DO A RELOCATION
    9720 045350  004767  000036            JSR     PC,TS12                 ;CHECK IF AN ABORT OCCURRED AND IF
    9721                                                                    ;YES IF MMR1 EQUALS EXPECTED DATA
    9722 045354  012703  012762            MOV     #12762,R3               ;SETUP EXPECTED DATA FOR MMR1
    9723 045360  012767  000001  135442    MOV     #1,FLAG                 ;SETUP FLAG FOR AN ABORT
    9724 045366  012737  000001  177572    MOV     #1,@#177572             ;TURN MMU ON
Z   9725 045374  010767  135412            MOV     R7,SLOC00               ;SAVE PC
    9726 045400  014225                    MOV     -(R2),(R5)+             ;DO A RELOCATION
    9727 045402  004767  000004            JSR     PC,TS12                 ;CHECK IF AN ABORT OCCURRED AND IF
    9728                                                                    ;YES IF MMR1 EQUALS EXPECTED DATA
    9729
    9730 045406  000167  000046            JMP     T12FIN
    9731                            ;
    9732                            ;ROUTINE TO CHECK IF AN ABORT OCCURRED AND IF MMR1 EQUALS EXPECTED DATA
    9733                            ;
    9734 045412  022767  000000  135410  TS12:   CMP     #0,FLAG                 ;DID AN ABORT OCCUR
    9735 045420  001401                    BEQ     1$                      ;YES GO ON
    9736 045422  104002                    ERROR   +2              ;MMU ERROR
    9737                                                           ;NO GO TO ERROR
    9738 045424  020367  135414          1$:     CMP     R3,SAVMR1               ; TEST MMR1 FOR EXPECTED DATA
```

MEMORY MANAGEMENT TESTS

```
 9739 045430  001401                        BEQ     2$                      ;OK GO ON
 9740 045432  104002                        ERROR   .+2                  ;MMU ERROR
 9741                                                                      ;NO GO TO ERROR
 9742 045434  026767  135352  135404  2$:   CMP     SLOC00,SAVMR2           ; TEST MMR2 FOR EXPECTED DATA
 9743 045442  001401                        BEQ     3$                      ;OK GO ON
 9744 045444  104002                        ERROR   .+2                  ;MMU ERROR
 9745                                                                      ;NO GO TO ERROR
 9746 045446  005067  135372          3$:   CLR     SAVMR1                  ;CLEAR STATUS REG SAVE AREA
 9747 045452  005067  135370                CLR     SAVMR2                  ;
 9748 045456  000207                        RTS     PC                      ;RETURN
 9749                                ;
 9750 045460  000240                T12FIN: NOP
 9760 045462                        TSMM13:                          ,
 9761                                ;       ADDER RELOCATION TEST PART A
 9762                                ;;*****************************************************************
                                    ;(NEED 16 BITS OF MEMORY ADDRESSING)
 9763 045462  005037  177572                CLR     @#177572                ;MMU OFF
 9764 045466  005067  135336                CLR     FLAG                    ;CLEAR MMU ABORT FLAG
 9765 045472  005037  177776                CLR     @#177776                ;INIT PSW
 9766 045476  004767  072050                JSR     PC,MMU                  ;INIT MMU
 9767 045502  012737  000020  172516        MOV     #20,@#172516            ;INIT MMR3
 9768 045510  012703  045664                MOV     #PARAD1,R3              ;SETUP PARS WITH TEST VALUES
 9769 045514  012701  045716                MOV     #PARVA1,R1              ;
 9770 045520  012133                1$:     MOV     (R1)+,@(R3)+            ;
 9771 045522  021127  000333                CMP     (R1),#333               ;
 9772 045526  001374                        BNE     1$                      ;
 9773 045530  012703  046002                MOV     #PHY1,R3                ;SET POINTERS TO ADDER PART A
 9774 045534  012701  045750                MOV     #VIR1,R1                ; TEST TABLES.
 9775 045540  012702  046034                MOV     #MODE1,R2               ;
 9776 045544  012237  177776          2$:   MOV     (R2)+,@#177776          ;INIT PSW
 9777 045550  013305                        MOV     @(R3)+,R5               ;SAVE DATA AT PHYSICAL ADDRESS
 9778 045552  012737  000001  177572        MOV     #1,@#177572             ;TURN MMU ON
 9779 045560  006531                        MFPI    @(R1)+                  ;SAVE DATA AT RELOCATED VIRTUAL ADDRESS
 9780 045562  012604                        MOV     (SP)+,R4                ;
 9781 045564  005037  177572                CLR     @#177572                ;TURN MMU OFF
 9782 045570  020504                        CMP     R5,R4                   ;IS DATA EQUAL TO EXPECTED
 9783 045572  001401                        BEQ     3$                      ;YES GO ON
 9784 045574  104002                        ERROR   .+2                  ;MMU ERROR
 9785                                                                      ;NO IT IS AN ADDER ERROR
 9786 045576  021327  000111          3$:   CMP     (R3),#111               ;ARE WE READY TO TEST DATA SPACE
 9787 045602  001360                        BNE     2$                      ;NO GO TO 2$
 9788 045604  005203                        INC     R3                      ;POINT TO DATA SPACE VALUES
 9789 045606  005203                        INC     R3                      ;
 9790 045610  005201                        INC     R1                      ;
 9791 045612  005201                        INC     R1                      ;
 9792 045614  005202                        INC     R2                      ;
 9793 045616  005202                        INC     R2                      ;
 9794 045620  012237  177776                MOV     (R2)+,@#177776          ;INIT PSW
 9795 045624  013305                        MOV     @(R3)+,R5               ;SAVE DATA AT PHYSICAL ADDRESS
 9796 045626  012737  000027  172516        MOV     #27,@#172516            ;INIT MMR3
 9797 045634  012737  000001  177572        MOV     #1,@#177572             ;TURN MMU ON
 9798 045642  106531                        MFPD    @(R1)+                  ;SAVE DATA AT RELOCATED VIRTUAL ADDRESS
 9799 045644  012604                        MOV     (SP)+,R4                ;POP THE STACK
 9800 045646  005037  177572                CLR     @#177572                ;TURN MMU OFF
 9801 045652  020504                        CMP     R5,R4                   ;IS DATA EQUAL TO EXPECTED
 9802 045654  001401                        BEQ     4$                      ;YES GO ON
 9803 045656  104002                        ERROR   .+2                  ;MMU ERROR
```

```
 9804                                                          ;NO IT IS AN ADDER ERROR
 9805 045660                                  4$:
 9806 045660  000167  000202                          JMP     T13FIN
 9807                                         ;
 9808                                         ;ADDER TEST PART A TABLES
 9809                                         ;
 9810 045664  172240                          PARAD1: .WORD   172240
 9811 045666  177642                                  .WORD   177642
 9812 045670  172252                                  .WORD   172252
 9813 045672  177640                                  .WORD   177640
 9814 045674  172242                                  .WORD   172242
 9815 045676  172254                                  .WORD   172254
 9816 045700  177652                                  .WORD   177652
 9817 045702  177644                                  .WORD   177644
 9818 045704  172246                                  .WORD   172246
 9819 045706  177654                                  .WORD   177654
 9820 045710  172250                                  .WORD   172250
 9821 045712  177660                                  .WORD   177660
 9822 045714  000333                                  .WORD   333
 9823 045716  000000                          PARVA1: .WORD   000000
 9824 045720  000010                                  .WORD   000010
 9825 045722  177777                                  .WORD   177777
 9826 045724  177601                                  .WORD   177601
 9827 045726  000010                                  .WORD   000010
 9828 045730  000052                                  .WORD   000052
 9829 045732  000070                                  .WORD   000070
 9830 045734  000010                                  .WORD   000010
 9831 045736  000010                                  .WORD   000010
 9832 045740  000060                                  .WORD   000060
 9833 045742  000000                                  .WORD   000000
 9834 045744  000010                                  .WORD   000010
 9835 045746  000333                                  .WORD   333
 9836 045750  000000                          VIR1:   .WORD   000000
 9837 045752  025000                                  .WORD   025000
 9838 045754  135224                                  .WORD   135224
 9839 045756  017700                                  .WORD   017700
 9840 045760  033000                                  .WORD   033000
 9841 045762  145252                                  .WORD   145252
 9842 045764  121000                                  .WORD   121000
 9843 045766  043000                                  .WORD   043000
 9844 045770  075000                                  .WORD   075000
 9845 045772  142000                                  .WORD   142000
 9846 045774  117700                                  .WORD   117700
 9847 045776  000111                                  .WORD   111
 9848 046000  007000                                  .WORD   007000
 9849 046002  000000                          PHY1:   .WORD   000000
 9850 046004  006000                                  .WORD   006000
 9851 046006  015124                                  .WORD   015124
 9852 046010  000000                                  .WORD   000000
 9853 046012  014000                                  .WORD   014000
 9854 046014  012452                                  .WORD   012452
 9855 046016  010000                                  .WORD   010000
 9856 046020  004000                                  .WORD   004000
 9857 046022  016000                                  .WORD   016000
 9858 046024  010000                                  .WORD   010000
 9859 046026  017700                                  .WORD   017700
 9860 046030  000111                                  .WORD   111
```

MEMORY MANAGEMENT TESTS

```
 9861 046032  010000                          .WORD   010000
 9862 046034  010000                MODE1:    .WORD   010000
 9863 046036  030000                          .WORD   030000
 9864 046040  010000                          .WORD   010000
 9865 046042  030000                          .WORD   030000
 9866 046044  010000                          .WORD   010000
 9867 046046  010000                          .WORD   010000
 9868 046050  030000                          .WORD   030000
 9869 046052  030000                          .WORD   030000
 9870 046054  010000                          .WORD   010000
 9871 046056  030000                          .WORD   030000
 9872 046060  010000                          .WORD   010000
 9873 046062  000111                          .WORD   111
 9874 046064  030000                          .WORD   030000
 9875                               ;
 9876 046066                        T13FIN:
 9887 046066                        TS1822:
 9888                               ;       TEST 22/18 BIT ADDRESS OPTION
 9889                               ;;*****************************************************************
                                    ;CHECK THE SOFTWARE SWITCH REGISTER TO DETERMINE IF THIS IS A 22 BIT OR AN
                                    ;18 BIT ADDRESS SYSTEM. BIT 08 IN THE SWR=1 INDICATES AN 18 BIT SYSTEM.
                                    ;IF WE'RE IN A 22 BIT SYSTEM WE CAN PERFORM SOME EXTRA TESTS.
 9890 046066  032777  000400  133044          BIT     #BIT08,@SWR             ;IS BIT 08 SET?
 9891 046074  001405                          BEQ     100$                    ;BRANCH IF ITS NOT
 9892 046076  062737  000001  001204          ADD     #1,@#$TESTN             ;KEEP TEST NUMBERS IN ORDER
 9893                                                                         ;ADD 1 FOR THE TESTS WE'RE SKIPPING
 9894 046104  000167  001422                  JMP     T14FIN                  ;SKIP OVER THESE TESTS IF IT IS
 9895                               ;
 9896                               ;IF THIS IS A 22 BIT SYSTEM CHECK THE 22/18 BIT ADDRESS OPTION
 9897                               ;
 9898                               ;TO TEST 22 BIT ADDRESSING WE DO THE FOLLOWING:
 9899                               ;       A. ENABLE 22 BIT ADDRESSING MODE
 9900                               ;       B. CLEAR ADDRESS 0
 9901                               ;       C. WRITE PHYSICAL ADDRESS 17000000 WITH ALL ONES
 9902                               ;       D. CHECK ADDRESS 0
 9903                               ;IF ADDRESS 0 IS UNCHANGED (=0) OR A TIME OUT OCCURRED, IT INDICATES
 9904                               ;22 BIT MODE IS FUNCTIONING.
 9905                               ;IF ADDRESS 0 =177777 IT INDICATES THAT 22 BIT MODE IS NOT FUNCTIONING.
 9906
 9907 046110  013767  000004  134674  100$:    MOV     @#4,SLOC00              ;SAVE VECTOR
 9908 046116  013767  000006  134670          MOV     @#6,SLOC01              ;SAVE VECTOR
 9909 046124  012737  046204  000004          MOV     #1$,@#4                 ;SET VECTOR FOR NXM TRAP
 9910 046132  012737  000340  000006          MOV     #340,@#6                ;
 9911 046140  012767  000020  124350          MOV     #20,SR3                 ;ENABLE 22 BIT MODE ADDRESSING
 9912 046146  012767  170000  124200          MOV     #170000,KIPAR6          ;SET KIPAR6 FOR 1920-1924KW ADDR RANGE
 9913 046154  012767  000001  131410          MOV     #1,SR0                  ;ENABLE MMU
 9914 046162  005067  131612                  CLR     0                       ;CLEAR ADDR 0
 9915 046166  012737  177777  140000          MOV     #177777,@#140000        ;MOVE ALL ONES TO ADDR 17000000 VIA KIPAR6
 9916                                                                         ;A TIME OUT ERROR OR
 9917 046174  005767  131600                  TST     0                       ;ADDR 0 REMAINING CLEAR INDICATES
 9918                                                                         ;THAT 22 BIT ADDRESS MODE IS WORKING AND
 9919                                                                         ;THAT SOME FURTHER TESTS SHOULD BE PERFORMED
 9920 046200  001405                          BEQ     2$                      ;IF ADDR 0 =177777
 9921                                                                         ;ERROR! 22 BIT ADDRESS MODE BAD
 9922 046202  104002                          ERROR   +2              ;MMU ERROR
 9923 046204  012706  001000          1$:     MOV     #STBOT,SP               ;GOT HERE AS A RESULT OF NXM TRAP---
 9924                                                                         ;CLEAN UP THE STACK
```

```
 9925 046210  005037  177766                 CLR     @#177766                ;CLEAR CPU ERROR REGISTER
 9926 046214  012737  046254  000004  2$:    MOV     #3$,@#4                 ;SET UP VECTOR FOR NXM TRAP
 9927 046222  042767  000020  124266         BIC     #BIT04,SR3              ;SET 18 BIT ADDRESSING MODE IN SR3
 9928 046230  012767  170000  124116         MOV     #170000,KIPAR6          ;SET KIPAR6 SO THAT BITS 18-21 SHOULD
 9929                                                                        ;BE ASSERTED IF 22 BIT ADR WAS ENABLED
 9930 046236  012737  177777  140000         MOV     #177777,@#140000        ;TRY TO WRITE ADDR 17000000 VIA KIPAR6
 9931                                                                        ;ADDR 0 SHOULD = 177777. A TIME OUT
 9932 046244  022737  177777  000000         CMP     #177777,@#0             ;OR ADDR 0 = ZERO INDICATES AN ERROR
 9933 046252  001403                         BEQ     4$                      ;GO TO NEXT TEST IF ADDR 0=177777
 9934 046254  005067  131312          3$:    CLR     SR0                     ;DISABLE MMU BEFORE ERROR.
 9935                                                                        ;ERROR! 18 BIT ADDR OPTION IS N.G.
 9936 046260  104002                         ERROR   +2              ;MMU ERROR
 9937                                 ;
 9938                                 ;TEST ADDRESS BITS 18 THRU 21
 9939                                 ;
 9940
 9941 046262  052767  000020  124226  4$:    BIS     #BIT04,SR3              ;ENABLE 22 BIT ADDRESSING MODE
 9942 046270  012767  046332  131506         MOV     #5$,4                   ;SET UP FOR NXM TRAP
 9943 046276  005067  131476                 CLR     0                       ;CLEAR ADDRESS 0
 9944 046302  012767  010000  124044         MOV     #10000,KIPAR6           ;TEST ADDRESS BIT 18
 9945 046310  012737  177777  140000         MOV     #177777,@#140000        ;WRITE ALL ONES TO ADDR 1000000
 9946 046316  005767  131456                 TST     0                       ;TEST ADDRESS 0. SHOULD = ZERO
 9947 046322  001403                         BEQ     5$                      ;BRANCH IF ADDRESS 0=0
 9948 046324  005067  131242                 CLR     SR0                     ;DISABLE MMU BEFORE ERROR
 9949 046330  104002                         ERROR   +2              ;MMU ERROR
 9950                                                                        ;ERROR! BIT 18 DID NOT ASSERT
 9951 046332  012737  046374  000004  5$:    MOV     #6$,@#4                 ;SET UP FOR NXM TRAP
 9952 046340  005067  131434                 CLR     0                       ;CLEAR ADDR 0
 9953 046344  012767  020000  124002         MOV     #20000,KIPAR6           ;TEST ADDRESS BIT 19
 9954 046352  012737  177777  140000         MOV     #177777,@#140000        ;WRITE ALL ONES TO ADDR 2000000
 9955 046360  005767  131414                 TST     0                       ;TEST ADDR 0. SHOULD= ZERO
 9956 046364  001403                         BEQ     6$                      ;BRANCH IF ADDRESS 0=0
 9957 046366  005067  131200                 CLR     SR0                     ;DISABLE MMU BEFORE ERROR
 9958 046372  104002                         ERROR   +2              ;MMU ERROR
 9959                                                                        ;ERROR! BIT 19 DID NOT ASSERT
 9960 046374  012737  046436  000004  6$:    MOV     #7$,@#4                 ;SET UP FOR NXM TRAP
 9961 046402  005067  131372                 CLR     0                       ;CLEAR ADDR 0
 9962 046406  012767  040000  123740         MOV     #40000,KIPAR6           ;TEST ADDRESS BIT 20
 9963 046414  012737  177777  140000         MOV     #177777,@#140000        ;WRITE ALL ONES TO ADDR 4000000
 9964 046422  005767  131352                 TST     0                       ;TEST ADDR 0. SHOULD =0
 9965 046426  001403                         BEQ     7$                      ;BRANCH IF ADDRESS 0 =0
 9966 046430  005067  131136                 CLR     SR0                     ;DISABLE MMU BEFORE ERROR
 9967 046434  104002                         ERROR   +2              ;MMU ERROR
 9968                                                                        ;ERROR! BIT 20 DID NOT ASSERT
 9969 046436  012737  046500  000004  7$:    MOV     #8$,@#4                 ;SET UP FOR NXM
 9970 046444  005067  131330                 CLR     0                       ;CLEAR ADDRESS 0
 9971 046450  012767  100000  123676         MOV     #100000,KIPAR6          ;TEST ADDRESS BIT 21
 9972 046456  012737  177777  140000         MOV     #177777,@#140000        ;WRITE ALL ONES AT ADDR 10000000
 9973 046464  005767  131310                 TST     0                       ;CHECK ADDRESS 0. SHOULD = 0
 9974 046470  001403                         BEQ     8$                      ;BRANCH IF ADDR 0 = 0
 9975 046472  005067  131074                 CLR     SR0                     ;DISABLE MMU    BEFORE ERROR
 9976 046476  104002                         ERROR   +2              ;MMU ERROR
 9977                                                                        ;ERROR! ADDR BIT 21 DID NOT ASSERT
 9978 046500  005067  131066          8$:    CLR     SR0                     ;DISABLE MMU
 9979 046504  005037  177766                 CLR     @#177766                ;CLEAR CPU ERROR REGISTER
 9980 046510  012706  001000                 MOV     #STBOT,R6               ;RESET STACK POINTER
 9981 046514  013737  003012  000004         MOV     @#SLOC00,@#4            ;RESTORE VECTORS
```

MEMORY MANAGEMENT TESTS

```
 9982 046522  013737  003014  000006          MOV     @#SLOC01,@#6            ;
 9983
 9992 046530                                TSMM14:
 9993                                       ;       ADDER RELOCATION TEST PART B
 9994                                       ;;*****************************************************************
                                            ;(NEED 22 BITS OF MEMORY ADDRESSING)
 9995 046530  005037  177572                        CLR     @#SR0                   ;TURN OFF MMU.
 9996 046534  005067  131226                        CLR     CPEREG                  ;CLEAR THE CPU ERROR REGISTER
 9997 046540  013737  000004  003012              MOV     @#4,@#SLOC00            ;SAVE LOC 4 IN SLOC00.
 9998 046546  013737  000006  003014              MOV     @#6,@#SLOC01            ;SAVE LOC 6 IN SLOC01.
 9999 046554  012737  047244  000004              MOV     #NXMTRP,@#4             ;SET UP FOR TIMEOUT TRAP
10000 046562  012737  000340  000006              MOV     #340,@#6                ;SET UP FOR TIMEOUT TRAP
10001 046570  005037  172340                      CLR     @#KIPAR0                ;SET KER PAR0 FOR 1ST 4KW OF MEMORY.
10002 046574  012767  077406  123476              MOV     #77406,KIPDR0           ;SET KER PDR FOR 4KW R/W ACCESS.
10003 046602  012737  177500  172354              MOV     #177500,@#KIPAR6        ;SET UP KERNEL PAGE ADDR REG 6
10004                                                                             ;FOR HIGHEST 4K WORDS OF NON-I/O
10005                                                                             ;FOR 2 MEG WORDS OF MEMORY.
10006 046610  012767  077406  123476              MOV     #77406,KIPDR6           ;SET KER PDR6 FOR 4KW R/W ACCESS.
10007 046616  012737  000020  172516              MOV     #20,@#SR3               ;ENABLE 22 BIT ADDRESSING.
10008 046624  012737  000001  177572              MOV     #1,@#SR0                ;TURN ON THE MMU.
10009 046632  005737  157776                      TST     @#157776                ;ATTEMPT TO ADDRESS LAST MEMORY ADDR.
10010                                      ;*************WILL TRAP TO 4 IF 2 MEG WORDS OF MEMORY NOT AVAILABLE*********
10011 046636  013737  003012  000004              MOV     @#SLOC00,@#4            ;RESTORE LOC 4
10012 046644  013737  003014  000006              MOV     @#SLOC01,@#6            ;RESTORE LOC 6
10013 046652  005037  177572                      CLR     @#177572                ;MMU OFF
10014 046656  005037  003030                      CLR     @#FLAG                  ;CLEAR MMU ABORT FLAG
10015 046662  004767  070664                      JSR     PC,MMU                  ;INIT MMU
10016 046666  012737  010000  177776              MOV     #10000,@#177776         ;INIT PSW
10017 046674  012737  000020  172516              MOV     #20,@#172516            ;INIT MMR3
10018 046702  052737  001000  177746              BIS     #1000,@#177746          ;TURN CACHE TEST FEATURE ON
10019 046710  012704  047446                      MOV     #PARVA3,R4              ;SET POINTERS TO INIT TABLES
10020 046714  012701  047500                      MOV     #VIR3,R1                ;
10021 046720  012437  172246          1$:         MOV     (R4)+,@#172246          ;INIT SIPAR3
10022 046724  012737  000001  177572              MOV     #1,@#177572             ;TURN MMU ON
10023 046732  012746  125252                      MOV     #125252,-(SP)           ;PUSH BACKGROUND DATA ON TO THE STACK
10024 046736  006671  000000                      MTPI    @0(R1)                  ;WRITE DATA TO PHYSICAL ADDRESS
10025 046742  006531                             MFPI    @(R1)+                  ;WRITE DATA AT PHYSICAL ADDRESS TO STACK
10026 046744  022726  125252                      CMP     #125252,(SP)+           ;IS DATA EQUAL TO EXPECTED
10027 046750  001403                             BEQ     2$                      ;YES GO ON
10028 046752  005037  177572                      CLR     @#177572                ;TURN MMU OFF
10029 046756  104002                             ERROR   +2                  ;MMU ERROR
10030                                                                       ;NOT EQUAL GO TO ERROR
10031 046760  005037  177572          2$:         CLR     @#177572                ;TURN MMU OFF
10032 046764  021427  000333                      CMP     (R4),#333               ;ARE WE DONE
10033 046770  001353                             BNE     1$                      ;NO GO TO 1$
10034 046772  012704  047330                      MOV     #PARVA2,R4              ;SET POINTERS TO PAR INIT TABLES
10035 046776  012701  047276                      MOV     #PARAD2,R1              ;
10036 047002  012431                  3$:         MOV     (R4)+,@(R1)+            ;INIT PARS
10037 047004  021127  000333                      CMP     (R1),#333               ;ARE WE DONE
10038 047010  001374                             BNE     3$                      ;NO, GO TO 3$
10039 047012  012704  047414                      MOV     #MODE2,R4               ;SET POINTERS TO ADDER PART B TABLES
10040 047016  012701  047362                      MOV     #VIR2,R1                ;
10041 047022  012702  047446                      MOV     #PARVA3,R2              ;
10042 047026  012703  047500                      MOV     #VIR3,R3                ;
10043 047032  004767  000076          4$:         JSR     PC,TS14                 ;WRITE DATA TO PHYSICAL ADDRESS AND THEN
10044                                                                             ;CHECK IF DATA AT PHYSICAL ADDRESS IS
10045                                                                             ;EQUAL TO EXPECTED AND IF NOT DETERMINE
```

MEMORY MANAGEMENT TESTS

```
10046
10047 047036  021127  000111                CMP     (R1),#111               ;IF IT IS AN ADDER ERROR OR A MEMORY ERROR
10048                                                                         ;HAVE WE DONE ALL THE 22 BIT MODE I SPACE
10049 047042  001373                        BNE     4$                      ;CASES
10050 047044  005201                        INC     R1                      ;NO GO TO 4$
10051 047046  005201                        INC     R1                      ;POINT TO 22 BIT MODE D SPACE CASE
10052 047050  005204                        INC     R4                      ;
10053 047052  005204                        INC     R4                      ;
10054 047054  012737  000027  172516        MOV     #27,@#172516            ;INIT MMR3
10055 047062  012437  177776                MOV     (R4)+,@#177776          ;INIT PSW
10056 047066  012746  052525                MOV     #52525,-(SP)            ;PUSH DATA ONTO STACK
10057 047072  012737  000001  177572        MOV     #1,@#177572             ;TURN MMU ON
10058 047100  106631                        MTPD    @(R1)+                  ;WRITE DATA TO PHYSICAL ADDRESS
10059 047102  005037  177572                CLR     @#177572                ;TURN MMU OFF
10060 047106  012737  000020  172516        MOV     #20,@#172516            ;INIT MMR3
10061 047114  004767  000040                JSR     PC,T14                  ;CHECK IF DATA AT PHYSICAL ADDRESS IS EQUAL
10062                                                                         ;TO EXPECTED AND IF NOT DETERMINE IF IT
10063                                                                         ;IS AN ADDER ERROR OR A MEMORY ERROR
10064 047120  005037  172516                CLR     @#172516                ;INIT MMR3 FOR 18 BIT MODE
10065 047124  004767  000004                JSR     PC,TS14                 ;WRITE DATA TO PHYSICAL ADDRESS AND THEN
10066                                                                         ;CHECK IF DATA AT PHYSICAL ADDRESS IS
10067                                                                         ;EQUAL TO EXPECTED AND IF NOT DETERMINE IF
10068                                                                         ;IT IS AN ADDER ERROR OR A MEMORY ERROR
10069
10070 047130  000167  000376                JMP     T14FIN
10071                                 ;
10072                                 ;ROUTINE TO WRITE DATA TO PHYSICAL ADDRESS AND TO CHECK IF DATA AT
10073                                 ;PHYSICAL ADDRESS IS EQUAL TO EXPECTED AND IF NOT DETERMINE IF IT IS
10074                                 ;AN ADDER ERROR OR A MEMORY ERROR
10075                                 ;
10076 047134  012437  177776          TS14:   MOV     (R4)+,@#177776          ;INIT PSW
10077 047140  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
10078 047146  012746  052525                  MOV     #52525,-(SP)            ;WRITE DATA ONTO STACK
10079 047152  006631                          MTPI    @(R1)+                  ;WRITE DATA TO PHYSICAL ADDRESS VIA STACK
10080 047154  005037  177572                  CLR     @#177572                ;TURN MMU OFF
10081 047160  012737  010000  177776  T14:    MOV     #10000,@#177776         ;INIT PSW
10082 047166  012237  172246                  MOV     (R2)+,@#172246          ;INIT SIPAR3
10083 047172  012737  000001  177572          MOV     #1,@#177572             ;TURN MMU ON
10084 047200  006573  000000                  MFPI    @0(R3)                  ;DO RELOCATION
10085 047204  022726  052525                  CMP     #52525,(SP)+            ;IS DATA EQUAL TO EXPECTED
10086 047210  001410                          BEQ     2$                      ;YES GO ON
10087 047212  006573  000000                  MFPI    @0(R3)                  ;WHAT TYPE OF ERROR IS IT
10088 047216  022726  125252                  CMP     #125252,(SP)+           ;
10089 047222  001402                          BEQ     1$                      ;
10090 047224  104002                          ERROR   .+2             ;MMU ERROR
10091                                                                 ;IT IS A MEMORY ERROR
10092 047226  000401                          BR      2$                      ;
10093 047230  104002                  1$:     ERROR   .+2             ;MMU ERROR
10094                                                                 ;IT IS AN ADDER ERROR
10095 047232  005037  177572          2$:     CLR     @#177572                ;TURN MMU OFF
10096 047236  005203                          INC     R3                      ;
10097 047240  005203                          INC     R3                      ;
10098 047242  000207                          RTS     PC                      ;RETURN
10099                                 ;
10100                                 ;NON-EXISTANT MEMORY TRAP ROUTINE
10101                                 ;
10102 047244  005037  177572          NXMTRP: CLR     @#SR0                   ;TURN OFF MMU.
```

MEMORY MANAGEMENT TESTS

```
10103 047250  012716  047532                    MOV     #T14FIN,(SP)          ;SET UP STACK WITH RETURN ADDR.
10104 047254  013737  003012  000004            MOV     @#SLOC00,@#4          ;RESTORE LOC 4
10105 047262  013737  003014  000006            MOV     @#SLOC01,@#6          ;RESTORE LOC 6
10106 047270  005037  177766                    CLR     @#177766              ;CLEAR TIME OUT INDICATION FROM
10107                                                                         ;CPU ERROR REGISTER.
10108 047274  000006                            RTT                           ;RETURN FROM TRAP; GO TO NEXT TEST.
10109                                 ;
10110                                 ;ADDER TEST PART B TABLES
10111                                 ;
10112 047276  177646                  PARAD2:   .WORD   177646
10113 047300  177650                            .WORD   177650
10114 047302  177652                            .WORD   177652
10115 047304  172240                            .WORD   172240
10116 047306  177640                            .WORD   177640
10117 047310  177642                            .WORD   177642
10118 047312  172244                            .WORD   172244
10119 047314  177644                            .WORD   177644
10120 047316  172252                            .WORD   172252
10121 047320  172352                            .WORD   172352
10122 047322  177662                            .WORD   177662
10123 047324  172242                            .WORD   172242
10124 047326  000333                            .WORD   333
10125 047330  157700                  PARVA2:   .WORD   157700
10126 047332  137700                            .WORD   137700
10127 047334  077700                            .WORD   077700
10128 047336  176777                            .WORD   176777
10129 047340  007600                            .WORD   007600
10130 047342  167700                            .WORD   167700
10131 047344  175700                            .WORD   175700
10132 047346  177425                            .WORD   177425
10133 047350  177220                            .WORD   177220
10134 047352  173700                            .WORD   173700
10135 047354  176700                            .WORD   176700
10136 047356  077400                            .WORD   077400
10137 047360  000333                            .WORD   333
10138 047362  070000                  VIR2:     .WORD   070000
10139 047364  110000                            .WORD   110000
10140 047366  130000                            .WORD   130000
10141 047370  000000                            .WORD   000000
10142 047372  000000                            .WORD   000000
10143 047374  030000                            .WORD   030000
10144 047376  050000                            .WORD   050000
10145 047400  052524                            .WORD   052524
10146 047402  136000                            .WORD   136000
10147 047404  130000                            .WORD   130000
10148 047406  000111                            .WORD   111
10149 047410  030000                            .WORD   030000
10150 047412  030000                            .WORD   030000
10151 047414  030000                  MODE2:    .WORD   030000
10152 047416  030000                            .WORD   030000
10153 047420  030000                            .WORD   030000
10154 047422  010000                            .WORD   010000
10155 047424  030000                            .WORD   030000
10156 047426  030000                            .WORD   030000
10157 047430  010000                            .WORD   010000
10158 047432  030000                            .WORD   030000
10159 047434  010000                            .WORD   010000
```

```
10160 047436  000000                          .WORD   000000
10161 047440  000111                          .WORD   111
10162 047442  030000                          .WORD   030000
10163 047444  010000                          .WORD   010000
10164 047446  160000                  PARVA3: .WORD   160000
10165 047450  140000                          .WORD   140000
10166 047452  100000                          .WORD   100000
10167 047454  176770                          .WORD   176770
10168 047456  007600                          .WORD   007600
10169 047460  170000                          .WORD   170000
10170 047462  176000                          .WORD   176000
10171 047464  177552                          .WORD   177552
10172 047466  177400                          .WORD   177400
10173 047470  174000                          .WORD   174000
10174 047472  177000                          .WORD   177000
10175 047474  007500                          .WORD   007500
10176 047476  000333                          .WORD   333
10177 047500  060000                  VIR3:   .WORD   060000
10178 047502  060000                          .WORD   060000
10179 047504  060000                          .WORD   060000
10180 047506  060700                          .WORD   060700
10181 047510  060000                          .WORD   060000
10182 047512  060000                          .WORD   060000
10183 047514  060000                          .WORD   060000
10184 047516  060024                          .WORD   060024
10185 047520  060000                          .WORD   060000
10186 047522  060000                          .WORD   060000
10187 047524  060000                          .WORD   060000
10188 047526  060000                          .WORD   060000
10189 047530  000333                          .WORD   333
10190
10191
10192                                 ;
10193 047532                          T14FIN:
10206
10207                                 ;       TEST NON-EXISTANT MEMORY TRAP
10208                                 ;;*****************************************************************
                                      ;WE ARE ASSUMING THAT THE NON-EXISTANT MEMORY TIME OUT
                                      ;FEATURE IS WORKING SINCE WE CAN'T GUARANTEE THAT
                                      ;THE SYSTEM BEING TESTED HAS A NON-EXISTANT MEMORY LOCATION.  .
                                      ;AT THIS TIME WE WILL ATTEMPT TO TEST THE NXM FUNCTION
10209 047532  004767  070014                  JSR     PC,MMU                  ;INIT THE MMU
10210 047536  012737  177400  172354          MOV     #177400,@#KIPAR6        ;SET KIPAR6 TO RELOCATE TO HIGHEST MEMORY
10211 047544  016767  130234  133240          MOV     4,SLOC00                ;SAVE VECTOR
10212 047552  016767  000026  130224          MOV     2$,4                    ;LOAD VEC WITH ADDR OF TRAP HANDLER
10213 047560  052767  000001  130004          BIS     #BIT00,SR0              ;TURN ON THE MMU
10214 047566  005067  130174                  CLR     CPEREG                  ;CLEAR THE CPU ERROR REGISTER
10215 047572  005067  130200                  CLR     PS                      ;CLEAR THE PSW
10216 047576  005737  157776                  TST     @#157776                ;ACCESS PHYSICAL ADDR 17757776
10217 047602  000415                  1$:     BR      NXMFIN                  ;IF IT DOESN'T TRAP WE'LL ASSUME
10218                                                                         ;THAT THIS IA A 4 MEGABYTE SYSTEM
10219                                                                         ;AND GO TO THE NEXT TEST
10220 047604  022767  000040  130154  2$:     CMP     #BIT05,CPEREG           ;IS CPU ERROR REGISTER CORRECT?
10221 047612  001401                          BEQ     3$                      ;
10222 047614  104002                          ERROR   .2              ;MMU ERROR
10223 047616  022726  047602          3$:     CMP     #1$,(SP)+               ;IS CONTENTS OF STACK CORRECT?
10224 047622  001401                          BEQ     4$                      ;
```

MEMORY MANAGEMENT TESTS

```
10225 047624  104002                        ERROR   +2              ;MMU ERROR
10226 047626  022726  000000          4$:   CMP     #0,(SP)+                ;IS CONTENTS OF STACK CORRECT?
10227 047632  001401                        BEQ     NXMFIN                  ;
10228 047634  104002                        ERROR   +2              ;MMU ERROR
10229 047636  005067  127730        NXMFIN: CLR     SR0                     ;TURN OFF THE MMU
10230 047642  005067  130120                CLR     CPEREG                  ;CLEAR THE CPU ERROR REGISTER
10231 047646  016767  133140  130130        MOV     SLOC00,4                ;RESTORE THE VECTOR
10232
10234
10236 047654                        TSMM15:
10237                               ;       PAGE WRITTEN BIT TEST
10238 047654  005037  177572                CLR     @#177572                ;MMU OFF
10239 047660  005067  133144                CLR     FLAG                    ;CLEAR MMU ABORT FLAG
10240 047664  004767  067662                JSR     PC,MMU                  ;INIT MMU
10241 047670  005037  177776                CLR     @#177776                ;INIT PSW
10242 047674  012704  172300                MOV     #172300,R4              ;SET POINTER TO KPDRS
10243 047700  004767  000114                JSR     PC,TS15                 ;DO RELOCATIONS AND TEST KIPDRS FOR
10244                                                                       ;PAGE WRITTEN BIT BEING SET AND IF
10245                                                                       ;NOT SET GO TO ERROR
10246 047704  004767  000160                JSR     PC,T15                  ;DO RELOCATIONS AND TEST KDPDRS FOR
10247                                                                       ;PAGE WRITTEN BIT BEING SET AND IF NOT
10248                                                                       ;SET GO TO ERROR
10249 047710  012737  050000  177776        MOV     #50000,@#177776         ;INIT PSW
10250 047716  012704  172200                MOV     #172200,R4              ;SET POINTER TO SPDRS
10251 047722  004767  000072                JSR     PC,TS15                 ;DO RELOCATIONS AND TEST SIPDRS FOR
10252                                                                       ;PAGE WRITTEN BIT BEING SET AND IF NOT
10253                                                                       ;SET GO TO ERROR
10254 047726  004767  000136                JSR     PC,T15                  ;DO RELOCATIONS AND TEST SDPDRS FOR
10255                                                                       ;PAGE WRITTEN BIT BEING SET AND IF NOT
10256                                                                       ;SET GO TO ERROR
10257 047732  005037  177776                CLR     @#177776                ;INIT PSW TO A KNOWN STATE
10258 047736  012737  170000  177776        MOV     #170000,@#177776        ;INIT PSW
10259 047744  012704  177600                MOV     #177600,R4              ;SET POINTER TO UPDRS
10260 047750  004767  000044                JSR     PC,TS15                 ;DO RELOCATIONS AND TEST UIPDRS FOR
10261                                                                       ;PAGE WRITTEN BIT BEING SET AND IF
10262                                                                       ;NOT SET GO TO ERROR
10263 047754  004767  000110                JSR     PC,T15                  ;DO RELOCATIONS AND TEST UDPDRS FOR
10264                                                                       ;PAGE WRITTEN BIT BEING SET AND IF NOT
10265                                                                       ;SET GO TO ERROR
10266 047760  005037  177776                CLR     @#177776                ;INIT PSW TO A KNOWN STATE
10267 047764  012704  172300                MOV     #172300,R4              ;SET POINTER TO KPDRS
10268 047770  004767  000152                JSR     PC,T15A                 ;EXPLICITLY WRITE TO KPDRS AND TEST
10269                                                                       ;FOR PAGE WRITTEN BIT BEING CLEARED
10270                                                                       ;AND IF NOT CLEARED GO TO ERROR
10271 047774  012704  172200                MOV     #172200,R4              ;SET POINTER TO SPDRS
10272 050000  004767  000142                JSR     PC,T15A                 ;EXPLICITLY WRITE TO SPDRS AND TEST
10273                                                                       ;FOR PAGE WRITTEN BIT BEING CLEARED
10274                                                                       ;AND IF NOT CLEARED GO TO ERROR
10275 050004  012704  177600                MOV     #177600,R4              ;SET POINTER TO UPDRS
10276 050010  004767  000132                JSR     PC,T15A                 ;EXPLICITLY WRITE TO UPDRS AND TEST
10277                                                                       ;FOR PAGE WRITTEN BIT BEING CLEARED
10278                                                                       ;AND IF NOT CLEARED GO TO ERROR
10279
10280 050014  000167  000154                JMP     T15FIN
10281                               ;
10282                               ;ROUTINE TO DO RELOCATIONS AND TEST IPDRS FOR PAGE WRITTEN BIT BEING
10283                               ;SET AND IF NOT SET REPORT AN ERROR
```

```
10284                                          ;
10285 050020  005001                    TS15:  CLR     R1                      ;SET POINTER TO VIRTUAL ADDRESS
10286 050022  012737  000020  172516           MOV     #20,@#172516            ;INIT MMR3
10287 050030  012737  000001  177572    1$:    MOV     #1,@#177572             ;TURN MMU ON
10288 050036  011111                           MOV     (R1),(R1)               ;DO A RELOCATION
10289 050040  005037  177572                   CLR     @#177572                ;TURN MMU OFF
10290 050044  022427  077506                   CMP     (R4)+,#77506            ;IS DATA EQUAL TO EXPECTED
10291 050050  001401                           BEQ     2$                      ;OK GO ON
10292 050052  104002                           ERROR   +2                    ;MMU ERROR
10293                                                                        ;NO GO TO ERROR
10294 050054  062701  020000              2$:  ADD     #20000,R1               ;POINT TO NEXT VIRTUAL ADDRESS
10295 050060  020127  160000                   CMP     R1,#160000              ;ARE WE DONE
10296 050064  001361                           BNE     1$                      ;NO GO TO 1$
10297 050066  000207                           RTS     PC                      ;RETURN
10298                                          ;
10299                                          ;ROUTINE TO DO RELOCATIONS AND TEST DPDRS FOR PAGE WRITTEN BIT BEING SET
10300                                          ;AND IF NOT SET REPORT AN ERROR
10301                                          ;
10302 050070  005001                    T15:   CLR     R1                      ;SET POINTER TO VIRTUAL ADDRESS
10303 050072  062704  000002                   ADD     #2,R4                   ;POINT TO FIRST DPDR
10304 050076  012737  000027  172516           MOV     #27,@#172516            ;INIT MMR3
10305 050104  012737  000001  177572    1$:    MOV     #1,@#177572             ;TURN MMU ON
10306 050112  011146                           MOV     (R1),-(SP)              ;PUSH DATA ONTO THE STACK
10307 050114  106611                           MTPD    (R1)                    ;DO A RELOCATION
10308 050116  005037  177572                   CLR     @#177572                ;TURN MMU OFF
10309 050122  022427  077506                   CMP     (R4)+,#77506            ;IS DATA EQUAL TO EXPECTED
10310 050126  001401                           BEQ     2$                      ;OK GO ON
10311 050130  104002                           ERROR   +2                    ;MMU ERROR
10312                                                                        ;NO GO TO ERROR
10313 050132  062701  020000              2$:  ADD     #20000,R1               ;POINT TO NEXT VIRTUAL ADDRESS
10314 050136  020127  160000                   CMP     R1,#160000              ;ARE WE DONE
10315 050142  001360                           BNE     1$                      ;NO GO TO 1$
10316 050144  000207                           RTS     PC                      ;RETURN
10317                                          ;
10318                                          ;ROUTINE TO EXPLICITLY WRITE TO PDRS AND TEST PAGE WRITTEN BIT FOR BEING
10319                                          ;CLEARED AND IF NOT CLEARED REPORT AN ERROR
10320                                          ;
10321 050146  005002                    T15A:  CLR     R2                      ;CLEAR COUNTER
10322 050150  011414                    1$:    MOV     (R4),(R4)               ;DO AN EXPLICIT WRITE TO PDR
10323 050152  022427  077406                   CMP     (R4)+,#77406            ;IS DATA EQUAL TO EXPECTED
10324 050156  001401                           BEQ     2$                      ;OK GO ON
10325 050160  104002                           ERROR   +2                    ;MMU ERROR
10326                                                                        ;NO GO TO ERROR
10327 050162  005202                    2$:    INC     R2                      ;INCREMENT POINTER
10328 050164  020227  000020                   CMP     R2,#20                  ;ARE WE DONE
10329 050170  001367                           BNE     1$                      ;NO GO TO 1$
10330 050172  000207                           RTS     PC                      ;RETURN
10331 050174                            T15FIN:
10332                                          ;
10335 050174                            TSMM16:
10336                                          ;       TEST CSM (CALL SUPERVISOR MODE)
10337 050174  005037  177572                   CLR     @#177572                ;MMU OFF
10338 050200  005037  003030                   CLR     @#FLAG                  ;CLEAR MMU ABORT FLAG
10339 050204  012704  050524                   MOV     #TMM16E,R4              ;INIT R4
10340 050210  004767  067336                   JSR     PC,MMU                  ;INIT MMU
10341 050214  012737  000037  172516           MOV     #37,@#172516            ;ENABLE CSM INSTRUCTION
10342 050222  005037  177776                   CLR     @#177776                ;SET PS TO KER MODE
```

MEMORY MANAGEMENT TESTS

```
10343 050226  013746  000010                    MOV     @#10,-(SP)              ;SAVE VECTORS
10344 050232  013746  000014                    MOV     @#14,-(SP)              ;
10345 050236  013746  000016                    MOV     @#16,-(SP)              ;
10346 050242  012737  050414  000010            MOV     #TMM16B,@#10            ;SETUP NEW VECTORS
10347 050250  012737  000137  000014            MOV     #137,@#14               ;
10348 050256  012737  050534  000016            MOV     #TMM16A,@#16            ;
10349 050264  007014                            .WORD   7014                    ; TEST INSTRUCTION
10350 050266  104002                            ERROR   .+2                   ;MMU ERROR
10351                                                                         ;GO TO ERROR IF NOT TRAPPED
10352 050270  012737  050444  000010  TSM16A:   MOV     #TMM16C,@#10            ;SETUP NEW VECTOR
10353 050276  012737  000027  172516            MOV     #27,@#172516            ;DISABLE CSM INSTRUCTION
10354 050304  012737  140000  177776            MOV     #140000,@#177776        ;SET PS TO USER MODE
10355 050312  007014                            .WORD   7014                    ; TEST INSTRUCTION
10356 050314  104002                            ERROR   .+2                   ;MMU ERROR
10357                                                                         ;GO TO ERROR IF NOT TRAPPED
10358 050316  012737  050474  000010  TSM16B:   MOV     #TMM16D,@#10            ;SETUP NEW VECTOR
10359 050324  012737  000027  172516            MOV     #27,@#172516            ;DISABLE CSM INSTRUCTION
10360 050332  005037  177776                    CLR     @#177776                ;SET PS TO KER MODE
10361 050336  007014                            .WORD   7014                    ; TEST INSTRUCTION
10362 050340  104002                            ERROR   .+2                   ;MMU ERROR
10363                                                                         ;GO TO ERROR IF NOT TRAPPED
10364 050342  012737  000037  172516  TSM16C:   MOV     #37,@#172516            ;ENABLE CSM INSTRUCTION
10365 050350  012737  040000  177776            MOV     #40000,@#177776         ;SET PS TO SUP MODE
10366 050356  012706  000700                    MOV     #700,R6                 ;INIT SUP SP
10367 050362  012737  140000  177776            MOV     #140000,@#177776        ;SET PS TO USER MODE
10368 050370  012706  000600                    MOV     #600,R6                 ;INIT USER SP
10369 050374  012737  000014  000010            MOV     #14,@#10                ;SETUP NEW VECTOR
10370 050402  000277                            SCC                             ;SET ALL CC BITS
10371 050404  007024                            .WORD   7024                    ; TEST INSTRUCTION
10372 050406  104002                  TSM16D:   ERROR   .+2                   ;MMU ERROR
10373                                                                         ;GO TO ERROR IF NOT TRAPPED
10374 050410  000167  000504                    JMP     TM16A
10375                                 ;
10376                                 ;
10377 050414  042737  007777  177776  TMM16B:   BIC     #7777,@#177776          ;CLEAR UNWANTED BITS
10378 050422  022737  000000  177776            CMP     #0,@#177776             ;IS PS CORRECT
10379 050430  001401                            BEQ     1$                      ;YES GO ON
10380 050432  104002                            ERROR   .+2                   ;MMU ERROR
10381                                                                         ;NO GO TO ERROR
10382 050434  005726                  1$:       TST     (SP)+                   ;CLEAN UP STACK
10383 050436  005726                            TST     (SP)+                   ;
10384 050440  000167  177624                    JMP     TSM16A                  ;CONTINUE TESTING
10385 050444  042737  007777  177776  TMM16C:   BIC     #7777,@#177776          ;CLEAR UNWANTED BITS
10386 050452  022737  030000  177776            CMP     #30000,@#177776         ;IS PS CORRECT
10387 050460  001401                            BEQ     1$                      ;YES GO ON
10388 050462  104002                            ERROR   .+2                   ;MMU ERROR
10389                                                                         ;NO GO TO ERROR
10390 050464  005726                  1$:       TST     (SP)+                   ;CLEAN UP STACK
10391 050466  005726                            TST     (SP)+                   ;
10392 050470  000167  177622                    JMP     TSM16B                  ;CONTINUE TESTING
10393 050474  042737  007777  177776  TMM16D:   BIC     #7777,@#177776          ;CLEAR UNWANTED BITS
10394 050502  022737  000000  177776            CMP     #0,@#177776             ;IS PS CORRECT
10395 050510  001401                            BEQ     1$                      ;YES GO ON
10396 050512  104002                            ERROR   .+2                   ;MMU ERROR
10397                                                                         ;NO GO TO ERROR
10398 050514  005726                  1$:       TST     (SP)+                   ;CLEAN UP STACK
10399 050516  005726                            TST     (SP)+                   ;
```

MEMORY MANAGEMENT TESTS

```
10400 050520  000167  177616                    JMP     TSM16C                  ;CONTINUE TESTING
10401 050524  156430                    TMM16E: .WORD   156430                  ; TEST LOCATION
10402 050526  104002                    TMM16F: ERROR   +2              ;MMU ERROR
10403                                           ;GO TO ERROR IF DIDN'T ABORT
10404 050530  000167  000364                    JMP     TM16A
10405 050534  022737  070017  177776    TMM16A: CMP     #70017,@#177776         ;IS PS CORRECT
10406 050542  001401                            BEQ     1$                      ;YES GO ON
10407 050544  104002                            ERROR   +2              ;MMU ERROR
10408                                           ;NO GO TO ERROR
10409 050546  020627  000572            1$:     CMP     R6,#572                 ;IS SP CORRECT
10410 050552  001401                            BEQ     2$                      ;YES GO ON
10411 050554  104002                            ERROR   +2              ;MMU ERROR
10412                                           ;NO GO TO ERROR
10413 050556  020427  050526            2$:     CMP     R4,#TMM16E+2            ;IS R4 CORRECT
10414 050562  001401                            BEQ     3$                      ;YES GO ON
10415 050564  104002                            ERROR   +2              ;MMU ERROR
10416                                           ;NO GO TO ERROR
10417 050566  023727  050524  156430    3$:     CMP     @#TMM16E,#156430        ;IS TEST LOCATION OK
10418 050574  001401                            BEQ     4$                      ;YES GO ON
10419 050576  104002                            ERROR   +2              ;MMU ERROR
10420                                           ;NO GO TO ERROR
10421 050600  022627  156430            4$:     CMP     (SP)+,#156430           ;IS STACK CORRECT
10422 050604  001401                            BEQ     5$                      ;YES GO ON
10423 050606  104002                            ERROR   +2              ;MMU ERROR
10424                                           ;NO GO TO ERROR
10425 050610  022627  050406            5$:     CMP     (SP)+,#TSM16D           ;IS STACK CORRECT
10426 050614  001401                            BEQ     6$                      ;YES GO ON
10427 050616  104002                            ERROR   +2              ;MMU ERROR
10428                                           ;NO GO TO ERROR
10429 050620  022627  140000            6$:     CMP     (SP)+,#140000           ;IS STACK CORRECT
10430 050624  001401                            BEQ     7$                      ;YES GO ON
10431 050626  104002                            ERROR   +2              ;MMU ERROR
10432                                           ;NO GO TO ERROR
10433 050630  012706  000700            7$:     MOV     #700,R6                 ;RESTORE SUP SP
10434 050634  012737  140000  177776            MOV     #140000,@#177776        ;SET PS TO USER MODE
10435 050642  020627  000600                    CMP     R6,#600                 ;IS USER SP CORRECT
10436 050646  001401                            BEQ     8$                      ;YES GO ON
10437 050650  104002                            ERROR   +2              ;MMU ERROR
10438                                           ;NO GO TO ERROR
10439 050652  012767  077400  121320    8$:     MOV     #77400,SIPDR0           ;SETUP SIPDR0 TO ABORT
10440 050660  012737  050526  000016            MOV     #TMM16F,@#16            ;SETUP VECTOR
10441 050666  012737  000001  003030            MOV     #1,@#FLAG               ;SETUP FLAG FOR AN ABORT
10442 050674  012737  000001  177572            MOV     #1,@#177572             ;TURN MMU ON
10443 050702  010701                            MOV     R7,R1                   ;SAVE OLD PC
10444 050704  007014                            .WORD   7014                    ; TEST INSTRUCTION
10445 050706  022737  000000  003030            CMP     #0,@#FLAG               ;DID AN ABORT OCCUR
10446 050714  001401                            BEQ     9$                      ;YES GO ON
10447 050716  104002                            ERROR   +2              ;MMU ERROR
10448                                           ;NO GO TO ERROR
10449 050720  023701  003046            9$:     CMP     @#SAVMR2,R1             ;IS MMR2 CORRECT
10450 050724  001401                            BEQ     10$                     ;YES GO ON
10451 050726  104002                            ERROR   +2              ;MMU ERROR
10452                                           ;NO GO TO ERROR
10453 050730  023727  003042  100041    10$:    CMP     @#SAVMR0,#100041        ;IS MMR0 CORRECT
10454 050736  001401                            BEQ     11$                     ;YES GO ON
10455 050740  104002                            ERROR   +2              ;MMU ERROR
10456                                           ;NO GO TO ERROR
```

MEMORY MANAGEMENT TESTS

```
10457 050742  012737  000037  172516  11$:    MOV     #37,@#172516            ;ENABLE CSM
10458 050750  012737  040000  177776          MOV     #40000,@#177776         ;SET PSW TO SUP
10459 050756  012706  000700                  MOV     #700,R6                 ;SETUP SUP SP
10460 050762  012737  140000  177776          MOV     #140000,@#177776        ;SET PSW TO USE
10461 050770  012706  000600                  MOV     #600,R6                 ;SETUP USE SP
10462 050774  012737  000014  000010          MOV     #14,@#10                ;SETUP NEW VECTOR
10463 051002  012737  051024  000016          MOV     #TS16,@#16              ;SETUP NEW VECTOR
10464 051010  000277                          SCC                             ;SET ALL CC BITS
10465 051012  007027                          .WORD   7027                    ;TEST INSTRUCTION
10466 051014  045712                          .WORD   45712
10467 051016  104002                  TS16A:  ERROR   +2              ;MMU ERROR
10468                                                                 ;GO TO ERROR IF DIDN'T TRAP
10469 051020  000167  000074                  JMP     TM16A
10470 051024  022737  070017  177776  TS16:   CMP     #70017,@#177776         ;IS PSW CORRECT
10471 051032  001401                          BEQ     200$                    ;YES GO ON
10472 051034  104002                          ERROR   +2              ;MMU ERROR
10473                                                                 ;NO GO TO ERROR
10474 051036  020627  000572          200$:   CMP     R6,#572                 ;IS SP CORRECT
10475 051042  001401                          BEQ     201$                    ;YES GO ON
10476 051044  104002                          ERROR   +2              ;MMU ERROR
10477                                                                 ;NO GO TO ERROR
10478 051046  022627  045712          201$:   CMP     (SP)+,#45712            ;IS STACK CORRECT
10479 051052  001401                          BEQ     202$                    ;YES GO ON
10480 051054  104002                          ERROR   +2              ;MMU ERROR
10481                                                                 ;NO GO TO ERROR
10482 051056  022627  051016          202$:   CMP     (SP)+,#TS16A            ;IS STACK CORRECT
10483 051062  001401                          BEQ     203$                    ;YES GO ON
10484 051064  104002                          ERROR   +2              ;MMU ERROR
10485                                                                 ;NO GO TO ERROR
10486 051066  022627  140000          203$:   CMP     (SP)+,#140000           ;IS STACK CORRECT
10487 051072  001401                          BEQ     204$                    ;YES GO ON
10488 051074  104002                          ERROR   +2              ;MMU ERROR
10489                                                                 ;NO GO TO ERROR
10490 051076  012706  000700          204$:   MOV     #700,R6                 ;RESTORE SUP SP
10491 051102  012737  140000  177776          MOV     #140000,@#177776        ;SET PSW TO USER MODE
10492 051110  020627  000600                  CMP     R6,#600                 ;IS USER SP CORRECT
10493 051114  001401                          BEQ     TM16A                   ;YES GO ON
10494 051116  104002                          ERROR   +2              ;MMU ERROR
10495                                                                 ;NO GO TO ERROR
10496 051120  005037  177776          TM16A:  CLR     @#177776                ;SET PS TO KER MODE
10497 051124  012637  000016                  MOV     (SP)+,@#16              ;RESTORE VECTORS
10498 051130  012637  000014                  MOV     (SP)+,@#14              ;
10499 051134  012637  000010                  MOV     (SP)+,@#10              ;
10500
10504                                 ;*******************************************************************************
10505                                 .SBTTL  FLOATING POINT TESTS
10506                                 ;*******************************************************************************
10507                                 ;                       BEGIN FLOATING POINT TESTING
10508                                 ;*******************************************************************************
10509                                 ;*******************************************************************************
10522 051140                          MBT1:
10523
10524                                 ;       FPP REGISTER BIT TESTS
10525                                 ;;*****************************************************************
                                      ;R5=FPP POINTER
                                      ;R1=TEMPORARY COUNTER
                                      ;R2=POINTER TO EXPECTED DATA
```

FLOATING POINT TESTS

```
                                        ;R3=POINTER TO RECEIVED DATA
                                        ;R4=ODD/EVEN COUNTER
 10526 051140  170011                           SETD
 10527 051142  005005                   MBT2:   CLR     R5                      ;SETUP FPP ACC POINTER
 10528 051144  012702  003100                     MOV     #BTEXP,R2               ;POINT TO TEST DATA
 10529 051150  012703  003110                     MOV     #BTRES,R3               ;POINT TO RECEIVED DATA
 10530 051154  170400                   MBT2A:  CLRD    AC0                     ;SETUP FPP REGISTER VALUES
 10531 051156  174012                           STD     AC0,(R2)                ;CLEAR EXPECTED VALUE
 10532 051160  005004                           CLR     R4
 10533 051162  170400                   BTG0:   CLRD    AC0                     ;SETUP FPP REGISTER VALUES
 10534 051164  170401                           CLRD    AC1
 10535 051166  170402                           CLRD    AC2
 10536 051170  170403                           CLRD    AC3
 10537 051172  170404                           CLRD    AC4
 10538 051174  170405                           CLRD    AC5
 10539
 10540 051176  010501                           MOV     R5,R1                   ;GET FPP AC NUMBER INTO R1
 10541 051200  070127  000014                     MUL     #14,R1                  ;ALLOW 10 LOCATIONS FOR OPERATION
 10542 051204  062701  051212                     ADD     #MAC0,R1                ;SETUP JMP LOCATION
 10543 051210  000111                           JMP     (R1)
 10544 051212  172467  131662           MAC0:   LDD     BTEXP,AC0               ;LOAD TEST DATA INTO TEST REGISTER
 10545 051216  174067  131666           MAC0A:  STD     AC0,BTRES               ;SAVE TEST RESULT
 10546 051222  000167  000074                     JMP     MACE                    ;GET OUT
 10547 051226  172567  131646           MAC1:   LDD     BTEXP,AC1               ;LOAD TEST DATA INTO TEST REGISTER
 10548 051232  174167  131652                     STD     AC1,BTRES               ;SAVE TEST RESULT
 10549 051236  000167  000060                     JMP     MACE                    ;GET OUT
 10550 051242  172667  131632           MAC2:   LDD     BTEXP,AC2               ;LOAD TEST DATA INTO TEST REGISTER
 10551 051246  174267  131636                     STD     AC2,BTRES               ;SAVE TEST RESULT
 10552 051252  000167  000044                     JMP     MACE                    ;GET OUT
 10553 051256  172767  131616           MAC3:   LDD     BTEXP,AC3               ;LOAD TEST DATA INTO TEST REGISTER
 10554 051262  174367  131622                     STD     AC3,BTRES               ;SAVE TEST RESULT
 10555 051266  000167  000030                     JMP     MACE                    ;GET OUT
 10556 051272  172467  131602           MAC4:   LDD     BTEXP,AC0
 10557 051276  174004                           STD     AC0,AC4                 ;LOAD TEST DATA INTO TEST REGISTER
 10558 051300  172404                           LDD     AC4,AC0                 ;SAVE TEST RESULT
 10559 051302  000167  177710                     JMP     MAC0A                   ;GET OUT
 10560 051306  172467  131566           MAC5:   LDD     BTEXP,AC0               ;LOAD TEST DATA INTO TEST XFER REGISTER
 10561 051312  174005                           STD     AC0,AC5                 ;LOAD TEST REGISTER
 10562 051314  172405                           LDD     AC5,AC0                 ;STORE RESULT INTO XFER FPP REGISTER
 10563 051316  000167  177674                     JMP     MAC0A                   ;GET OUT
 10564 051322  026767  131552  131560   MACE:   CMP     BTEXP,BTRES
 10565 051330  001014                           BNE     BTER                    ;BRANCH IF REGISTER ERROR
 10566 051332  026767  131544  131552           CMP     BTEXP+2,BTRES+2
 10567 051340  001010                           BNE     BTER
 10568 051342  026767  131536  131544           CMP     BTEXP+4,BTRES+4
 10569 051350  001004                           BNE     BTER
 10570 051352  026767  131530  131536           CMP     BTEXP+6,BTRES+6
 10571 051360  001403                           BEQ     MBT8                    ;GOOD RESULT
 10572 051362  004737  140132           BTER:   CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
 10573 051366  104003                           ERROR   +3              ;FPP ERROR
 10574                                                                  ;FPP AC LOADED INCORRECTLY
 10575                                  ;NOW VERIFY THE OTHER REGISTERS REMAINED ZERO
 10576 051370                           MBT8:
 10577 051370  005001                           CLR     R1                      ;CLEAR TEMPORARY COUNTER
 10578 051372  005705                           TST     R5                      ;SEE IF R0 UNDER TEST
 10579 051374  001413                           BEQ     MBT8A                   ;BRANCH IF TEST ING R0
 10580 051376  020527  000004                     CMP     R5,#4                   ;SEE IF TESTING FPP REGISTER >R4
```

FLOATING POINT TESTS

```
10581 051402  100010                    BPL     MBT8A           ;SKIP R0 TESTING
10582 051404  174067  131500            STD     AC0,BTRES       ;SAVE AC TEST RESULT
10583 051410  004767  000246            JSR     R7,BTTST        ;VERIFY THAT CONTENTS REMAINED ZERO
10584 051414  001403                    BEQ     MBT8A           ;BRANCH IF EXPECTED RESULT
10585 051416  004737  140132            CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10586 051422  104003                    ERROR   +3              ;FPP ERROR
10587                                                           ;BAD AC0
10588 051424  020527  000001  MBT8A:    CMP     R5,#1           ;SEE IF R1 UNDER TEST
10589 051430  001410                    BEQ     MBT8B           ;BRANCH IF R1 UNDER TEST
10590 051432  174167  131452            STD     AC1,BTRES       ;SAVE AC TEST RESULT
10591 051436  004767  000220            JSR     R7,BTTST        ;VERIFY THAT CONTENTS REMAINED ZERO
10592 051442  001403                    BEQ     MBT8B           ;BRANCH IF GOOD
10593 051444  004737  140132            CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10594 051450  104003                    ERROR   +3              ;FPP ERROR
10595                                                           ;BAD AC1
10596 051452  020527  000002  MBT8B:    CMP     R5,#2           ;SEE IF TESTING FPP REGISTER AC2
10597 051456  001410                    BEQ     MBT8C           ;BRANCH IF R2 UNDER TEST
10598 051460  174267  131424            STD     AC2,BTRES       ;SAVE AC TEST RESULT
10599 051464  004767  000172            JSR     R7,BTTST        ;VERIFY THAT CONTENTS REMAINED ZERO
10600 051470  001403                    BEQ     MBT8C           ;BRANCH IF GOOD
10601 051472  004737  140132            CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10602 051476  104003                    ERROR   +3              ;FPP ERROR
10603                                                           ;BAD AC2
10604 051500  020527  000003  MBT8C:    CMP     R5,#3           ;SEE IF R3 UNDER TEST
10605 051504  001410                    BEQ     MBT8D           ;BRANCH IF R3 UNDER TEST
10606 051506  174367  131376            STD     AC3,BTRES       ;SAVE AC TEST RESULT
10607 051512  004767  000144            JSR     R7,BTTST        ;VERIFY THAT CONTENTS REMAINED ZERO
10608 051516  001403                    BEQ     MBT8D           ;BRANCH IF GOOD
10609 051520  004737  140132            CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10610 051524  104003                    ERROR   +3              ;FPP ERROR
10611                                                           ;BAD AC3
10612 051526  020527  000004  MBT8D:    CMP     R5,#4           ;SEE IF R4 UNDER TEST
10613 051532  001411                    BEQ     MBT8E           ;BRANCH IF R4 UNDER TEST
10614 051534  172404                    LDD     AC4,AC0         ;MOVE REGISTER CONTENT
10615 051536  174067  131346            STD     AC0,BTRES       ;SAVE AC TEST RESULT
10616 051542  004767  000114            JSR     R7,BTTST        ;VERIFY THAT CONTENTS REMAINED ZERO
10617 051546  001403                    BEQ     MBT8E           ;BRANCH IF GOOD
10618 051550  004737  140132            CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10619 051554  104003                    ERROR   +3              ;FPP ERROR
10620                                                           ;BAD AC4
10621 051556  020527  000005  MBT8E:    CMP     R5,#5           ;SEE IF R0 UNDER TEST
10622 051562  001411                    BEQ     MBT8F           ;BRANCH IF R0 UNDER TEST
10623 051564  172405                    LDD     AC5,AC0         ;MOVE REGISTER CONTENTS
10624 051566  174067  131316            STD     AC0,BTRES       ;SAVE AC TEST RESULT
10625 051572  004767  000064            JSR     R7,BTTST        ;VERIFY THAT CONTENTS REMAINED ZERO
10626 051576  001403                    BEQ     MBT8F           ;BRANCH IF GOOD
10627 051600  004737  140132            CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10628 051604  104003                    ERROR   +3              ;FPP ERROR
10629                                                           ;BAD AC5
10630 051606  005204  MBT8F:    INC     R4              ;INCREMENT PATTERN COUNTER
10631 051610  000241                    CLC
10632 051612  042704  177776            BIC     #177776,R4      ; TEST FOR ODD /EVEN
10633 051616  001401                    BEQ     MBT8FG          ;BRANCH IF EVEN
10634 051620  000261                    SEC                     ;SET CARRY FOR TEST PATTERN SHIFT
10635 051622  006112  MBT8FG:  ROL     (R2)            ;ROTATE LSW OF TEST PATTERN
10636 051624  006162  000002            ROL     2(R2)           ;ROTATE 2 WORD OF TEST PATTERN
10637 051630  006162  000004            ROL     4(R2)           ;ROTATE 3 WORD OF TEST PATTERN
```

FLOATING POINT TESTS

```
10638 051634  006162  000006                ROL     6(R2)                   ;ROTATE 4 WORD OF TEST PATTERN
10639 051640  103402                        BCS     MBT8I                   ;JUMP IF THROUGH WITH TEST PATTERN
10640 051642  000167  177314                JMP     BTGO                    ;CONTINUE WITH NEW TEST PATTERN
10641
10642 051646  005205                MBT8I:  INC     R5              ;GO TO NEXT REGISTER TEST
10643 051650  020527  000006                CMP     R5,#6           ;SEE IF THROUGH TESTING
10644 051654  100016                        BPL     MBTE            ;JUMP IF THROUGH
10645 051656  000167  177272                JMP     MBT2A           ;CONTINUE TESTING WITH NEW PATTERN
10646                               ;
10647                               ;
10648 051662  005767  131222        BTTST:  TST     BTRES           ;VERIFY CONTENTS AS ZERO
10649 051666  001010                        BNE     BTTSTE          ;EXIT IF NOT ZERO
10650 051670  005767  131216                TST     BTRES+2         ;VERIFY CONTENTS AS ZERO
10651 051674  001005                        BNE     BTTSTE          ;EXIT IF NOT ZERO
10652 051676  005767  131212                TST     BTRES+4         ;VERIFY CONTENTS AS ZERO
10653 051702  001002                        BNE     BTTSTE          ;EXIT IF NOT ZERO
10654 051704  005767  131206                TST     BTRES+6         ;VERIFY CONTENTS AS ZERO
10655 051710  000207                BTTSTE: RTS     R7              ;GO BACK TO CALLING ROUTINE
10656
10657
10658                                       ;
10659                                       ;
10660 051712                        MBTE:
10661
10663                               ;
10674 051712                        MFACU:
10675
10676                               ;       TEST UNIQUENESS OF FPP ACCUMULATORS
10677                               ;;*****************************************************************
                                    ;THIS TEST LOADS UNIQUE PATTERNS INTO EACH ACCUMULATOR SIMULTANEOUSLY.
                                    ;R2=POINTER TO EXPECTED DATA
                                    ;R3=POINTER TO RECEIVED DATA
10678                                       ;
10679 051712  170011                        SETD
10680 051714  005000                MFA:    CLR     R0                      ;SETUP FPP ACC POINTER
10681 051716  005004                        CLR     R4
10682 051720  012702  003100                MOV     #BTEXP,R2               ;POINT TO TEST DATA
10683 051724  012703  003110                MOV     #BTRES,R3               ;POINT TO RECEIVED DATA
10684 051730  012767  000051  131142        MOV     #51,BTEXP               ;SETUP EXPECTED DATA
10685 051736  012767  000052  131136        MOV     #52,BTEXP+2             ;
10686 051744  012767  000053  131132        MOV     #53,BTEXP+4             ;
10687 051752  012767  000054  131126        MOV     #54,BTEXP+6             ;
10688 051760  172467  131114                LDD     BTEXP,AC0               ;MOVE DATA TEMPORARILY
10689 051764  174005                        STD     AC0,AC5                 ;PUT DATA INTO TEST REGISTER
10690 051766  004567  000240                JSR     R5,SUBT                 ;SUBTRACT TEN FROM EACH EXPECTED DATA
10691 051772  172467  131102                LDD     BTEXP,AC0               ;MOVE DATA TEMPORARILY
10692 051776  174004                        STD     AC0,AC4                 ;MOVE DATA INTO TEST REGISTER
10693 052000  004567  000226                JSR     R5,SUBT                 ;SUBTRACT 10 FROM TEST DATA WORDS
10694 052004  172767  131070                LDD     BTEXP,AC3               ;STORE INTO TEST REGISTER
10695 052010  004567  000216                JSR     R5,SUBT                 ;GET NEXT SET OF UNIQUE DATA WORDS
10696 052014  172667  131060                LDD     BTEXP,AC2               ;STORE INTO TEST REGISTER
10697 052020  004567  000206                JSR     R5,SUBT                 ;GET NEXT SET OF TEST DATAS
10698 052024  172567  131050                LDD     BTEXP,AC1               ;LOAD TEST REGISTER
10699 052030  004567  000176                JSR     R5,SUBT                 ;GET NEXT SET OF TEST WORDS
10700 052034  172467  131040                LDD     BTEXP,AC0               ;LOAD FINAL TEST REGISTER
10701                                                                       ;ALL REGISTER CONTAIN UNIQUE TEST WORDS
10702 052040  174067  131044                STD     AC0,BTRES               ;STORE AC0,RESULT
```

FLOATING POINT TESTS

```
10703 052044  004567  000246              JSR     R5,BFA          ;CHECK RESULT
10704 052050  001403                      BEQ     BFAC1           ;BRANCH IF GOOD
10705 052052  004737  140132              CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10706 052056  104003                      ERROR   +3          ;FPP ERROR
10707                                     ;BAD AC0
10708 052060  004567  000200    BFAC1:    JSR     R5,ADDT         ;UPDATE EXPECTED RESULT
10709 052064  174167  131020              STD     AC1,BTRES       ;STORE AC1 RESULT
10710 052070  004567  000222              JSR     R5,BFA          ;CHECK RESULT
10711 052074  001403                      BEQ     BFAC2           ;BRANCH IF GOOD
10712 052076  004737  140132              CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10713 052102  104003                      ERROR   +3          ;FPP ERROR
10714                                     ;BAD RESULT AC1
10715 052104  004567  000154    BFAC2:    JSR     R5,ADDT         ;UPDATE EXPECTED RESULT
10716 052110  174267  130774              STD     AC2,BTRES       ;STORE AC2 RESULT
10717 052114  004567  000176              JSR     R5,BFA          ;CHECK RESULT
10718 052120  001403                      BEQ     BFAC3           ;BRANCH IF GOOD
10719 052122  004737  140132              CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10720 052126  104003                      ERROR   +3          ;FPP ERROR
10721                                     ;BAD AC2 RESLUT
10722 052130  004567  000130    BFAC3:    JSR     R5,ADDT         ;UPDATE EXPECTED RESULT
10723 052134  174367  130750              STD     AC3,BTRES       ;SAVE TEST RESULT
10724 052140  004567  000152              JSR     R5,BFA          ;CHECK RESULT
10725 052144  001403                      BEQ     BFAC4           ;BRANCH IF GOOD
10726 052146  004737  140132              CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10727 052152  104003                      ERROR   +3          ;FPP ERROR
10728                                     ;BAD AC3 RESULT
10729 052154  004567  000104    BFAC4:    JSR     R5,ADDT         ;UPDATE EXPECTED RESULT
10730 052160  172704                      LDD     AC4,AC3         ;SAVE TEMPORARY
10731 052162  174367  130722              STD     AC3,BTRES       ;STORE AC4 RESULT
10732 052166  004567  000124              JSR     R5,BFA          ;CHECK RESULT
10733 052172  001403                      BEQ     BFAC5           ;BRANCH IF GOOD
10734 052174  004737  140132              CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10735 052200  104003                      ERROR   +3          ;FPP ERROR
10736                                     ;BAD AC4 RESULT
10737 052202  004567  000056    BFAC5:    JSR     R5,ADDT         ;UPDATE EXPECTED RESULT
10738 052206  172605                      LDD     AC5,AC2         ;SAVE TEMPORARY COPY
10739 052210  174267  130674              STD     AC2,BTRES       ;MOVE AC5 RESULT
10740 052214  004567  000076              JSR     R5,BFA          ;CHECK RESULT
10741 052220  001456                      BEQ     BFAE            ;BRANCH IF GOOD
10742 052222  004737  140132              CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
10743 052226  104003                      ERROR   +3          ;FPP ERROR
10744                                     ;BAD AC5 RESULT
10745 052230  000452                      BR      BFAE            ;EXIT MODULE
10746
10747 052232  162767  000010  130640  SUBT:   SUB     #10,BTEXP               ;UPDATE EXPECTED CONTENTS
10748 052240  162767  000010  130634          SUB     #10,BTEXP+2             ;UPDATE EXPECTED CONTENTS
10749 052246  162767  000010  130630          SUB     #10,BTEXP+4             ;UPDATE EXPECTED CONTENTS
10750 052254  162767  000010  130624          SUB     #10,BTEXP+6             ;UPDATE EXPECTED CONTENTS
10751 052262  000205                      RTS     R5                  ;
10752 052264  062767  000010  130606  ADDT:   ADD     #10,BTEXP               ;UPDATE EXPECTED CONTENTS
10753 052272  062767  000010  130602          ADD     #10,BTEXP+2             ;UPDATE EXPECTED CONTENTS
10754 052300  062767  000010  130576          ADD     #10,BTEXP+4             ;UPDATE EXPECTED CONTENTS
10755 052306  062767  000010  130572          ADD     #10,BTEXP+6             ;UPDATE EXPECTED CONTENTS
10756 052314  000205                      RTS     R5                  ;
10757
10758 052316  026767  130556  130564  BFA:    CMP     BTEXP,BTRES         ;VERIFY CONTENTS
10759 052324  001013                      BNE     BFB                 ;EXIT IF NOT ZERO
```

FLOATING POINT TESTS

```
10760 052326  026767  130550  130556          CMP     BTEXP+2,BTRES+2         ;VERIFY CONTENTS
10761 052334  001007                          BNE     BFB                     ;EXIT IF NOT ZERO
10762 052336  026767  130542  130550          CMP     BTEXP+4,BTRES+4         ;VERIFY CONTENTS
10763 052344  001003                          BNE     BFB                     ;EXIT IF NOT ZERO
10764 052346  026767  130534  130542          CMP     BTEXP+6,BTRES+6         ;VERIFY CONTENTS
10765 052354  000205                  BFB:    RTS     R5                      ;GO BACK TO CALLING ROUTINE
10766
10767
10768 052356                          BFAE:
10769
10770
10771
10774 052356                          TSFP1:
10775                                 ;       TEST LDFPS AND STFPS MODE 0
10776 052356  005037  140054                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10777 052362  012704  147757                  MOV     #147757,R4              ;SETUP DATA TO BE LOADED
10778 052366  004767  000032                  JSR     PC,LOST                 ;LOAD AND STORE FPS WITH DATA
10779 052372  012704  105252                  MOV     #105252,R4              ;SETUP DATA TO BE LOADED
10780 052376  004767  000022                  JSR     PC,LOST                 ;LOAD AND STORE FPS WITH DATA
10781 052402  012704  042505                  MOV     #42505,R4               ;SETUP DATA TO BE LOADED
10782 052406  004767  000012                  JSR     PC,LOST                 ;LOAD AND STORE FPS WITH DATA
10783 052412  005004                          CLR     R4                      ;SETUP DATA TO BE LOADED
10784 052414  004767  000004                  JSR     PC,LOST                 ;LOAD AND STORE FPS WITH DATA
10785
10786                                 ;
10787 052420  000167  000020                  JMP     FIN1
10788                                 ;
10789 052424  170104                  LOST:   LDFPS   R4                      ;LOAD FPS WITH DATA
10790 052426  170201                          STFPS   R1                      ;LOAD R1 WITH (FPS)
10791 052430  020401                          CMP     R4,R1                   ;DID THE INSTRUCTIONS WORK
10792 052432  001403                          BEQ     1$                      ;YES GO ON
10793 052434  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10794 052440  104003                          ERROR   +3                      ;FPP ERROR
10795                                                                         ;NO GO TO ERROR
10796 052442  000207                  1$:     RTS     PC                      ;RETURN
10797 052444  000240                  FIN1:   NOP
10798                                 ;
10801 052446                          TSFP2:
10802                                 ;       TEST CFCC
10803 052446  005037  140054                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10804 052452  012704  000017                  MOV     #17,R4                  ;SETUP DATA TO BE LOADED
10805 052456  004767  000032                  JSR     PC,TSF2                 ;LOAD FPS AND COPY CONDITION CODES TO PS
10806 052462  012704  000012                  MOV     #12,R4                  ;SETUP DATA TO BE LOADED
10807 052466  004767  000022                  JSR     PC,TSF2                 ;LOAD FPS AND COPY CONDITION CODES TO PS
10808 052472  012704  000005                  MOV     #5,R4                   ;SETUP DATA TO BE LOADED
10809 052476  004767  000012                  JSR     PC,TSF2                 ;LOAD FPS AND COPY CONDITION CODES TO PS
10810 052502  005004                          CLR     R4                      ;SETUP DATA TO BE LOADED
10811 052504  004767  000004                  JSR     PC,TSF2                 ;LOAD FPS AND COPY CONDITION CODES TO PS
10812                                 ;
10813 052510  000167  000030                  JMP     FIN2
10814                                 ;
10815 052514  170104                  TSF2:   LDFPS   R4                      ;LOAD FPS
10816 052516  170000                          CFCC                            ;COPY CONDITION CODES TO PS
10817 052520  013701  177776                  MOV     @#177776,R1             ;SAVE PS TO R1
10818 052524  042701  177760                  BIC     #177760,R1              ;MASK OUT UNWANTED BITS
10819 052530  020401                          CMP     R4,R1                   ;WAS CONDITION CODE BITS TRANSFERRED
10820                                                                         ;CORRECTLY
```

FLOATING POINT TESTS

```
10821 052532  001403                    BEQ     1$                      ;YES GO ON
10822 052534  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10823 052540  104003                    ERROR   +3              ;FPP ERROR
10824                                                           ;NO GO TO ERROR
10825 052542  000207            1$:     RTS     PC                      ;RETURN
10826 052544                    FIN2:
10827                           ;
10830 052544                    TSFP3:
10831                           ;       TEST SETF, SETD, SETI, SETL
10832 052544  005037  140054              CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10833 052550  012704  000200              MOV     #200,R4                 ;SETUP DATA TO BE LOADED
10834 052554  170104                      LDFPS   R4                      ;LOAD FPS
10835 052556  170001                      SETF                            ;MAKE FD=0
10836 052560  170201                      STFPS   R1                      ;STORE FPS
10837 052562  020127  000000              CMP     R1,#0                   ;IS FD=0
10838 052566  001403                      BEQ     1$                      ;YES GO ON
10839 052570  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10840 052574  104003                    ERROR   +3              ;FPP ERROR
10841                                                           ;NO GO TO ERROR
10842 052576  170011            1$:     SETD                            ;MAKE FD=1
10843 052600  170201                      STFPS   R1                      ;STORE FPS
10844 052602  020104                      CMP     R1,R4                   ;IS FD=1
10845 052604  001403                      BEQ     2$                      ;YES GO ON
10846 052606  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10847 052612  104003                    ERROR   +3              ;FPP ERROR
10848                                                           ;NO GO TO ERROR
10849 052614  012704  000100      2$:     MOV     #100,R4                 ;SETUP DATA TO BE LOADED
10850 052620  170104                      LDFPS   R4                      ;LOAD FPS
10851 052622  170002                      SETI                            ;MAKE FL=0
10852 052624  170201                      STFPS   R1                      ;STORE FPS
10853 052626  020127  000000              CMP     R1,#0                   ;IS FL=0
10854 052632  001403                      BEQ     3$                      ;YES GO ON
10855 052634  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10856 052640  104003                    ERROR   +3              ;FPP ERROR
10857                                                           ;NO GO TO ERROR
10858 052642  170012            3$:     SETL                            ;MAKE FL=1
10859 052644  170201                      STFPS   R1                      ;STORE FPS
10860 052646  020104                      CMP     R1,R4                   ;IS FL=1
10861 052650  001403                      BEQ     4$                      ;YES GO ON
10862 052652  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10863 052656  104003                    ERROR   +3              ;FPP ERROR
10864                                                           ;NO GO TO ERROR
10865 052660                    4$:
10866                           ;
10869 052660                    TSFP4:
10870                           ;       TEST ILLEGAL OP CODES AND STST
10871 052660  005037  140054              CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10872 052664  012705  170003              MOV     #170003,R5              ;INIT OP CODE
10873 052670  013746  000244              MOV     @#244,-(SP)             ;SAVE FP VECTOR
10874 052674  012737  053030  000244      MOV     #ILLOP1,@#244           ;SETUP NEW VECTOR
10875 052702  013746  000004              MOV     @#4,-(SP)               ;SAVE TIME OUT VECTOR
10876 052706  012737  053114  000004      MOV     #TIMEOU,@#4             ;SETUP NEW VECTOR
10877 052714  013746  000010              MOV     @#10,-(SP)              ;SAVE ILLEGAL VECTOR
10878 052720  012737  053124  000010      MOV     #ILLOP2,@#10            ;SETUP NEW VECTOR
10879 052726  005003            D1:     CLR     R3                      ;
10880 052730  170103                      LDFPS   R3                      ;CLEAR FPS
10881 052732  005002                      CLR     R2                      ;
```

FLOATING POINT TESTS

```
10882 052734  010537  052740            MOV     R5,@#D2                 ;SETUP THE ILLEGAL INST
10883 052740  000000            D2:     .WORD   0                       ;
10884 052742  170000            D3:     CFCC                            ;MEMORY WORDS TO BE USED WITH
10885 052744  005202                    INC     R2                      ;EXECUTION OF ILLEGAL OP CODE
10886 052746  005202                    INC     R2                      ;
10887 052750  170201                    STFPS   R1                      ;SAVE FPS
10888 052752  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10889 052756  104003                    ERROR   .+3             ;FPP ERROR
10890                                                           ;GO TO ERROR
10891 052760  022705  170010    D4:     CMP     #170010,R5              ;COMPUTE NEXT OP CODE
10892 052764  001003                    BNE     D5                      ;
10893 052766  012705  170013            MOV     #170013,R5              ;
10894 052772  000755                    BR      D1                      ;
10895 052774  022705  170077    D5:     CMP     #170077,R5              ;
10896 053000  001001                    BNE     D6                      ;
10897 053002  000402                    BR      D7                      ;
10898 053004  005205            D6:     INC     R5                      ;
10899 053006  000747                    BR      D1                      ;
10900 053010  012637  000010    D7:     MOV     (SP)+,@#10              ;RESTORE VECTORS
10901 053014  012637  000004            MOV     (SP)+,@#4               ;
10902 053020  012637  000244            MOV     (SP)+,@#244             ;
10903
10904                           ;
10905 053024  000167  000104            JMP     FIN4
10906                           ;
10907 053030  022716  052742    ILLOP1: CMP     #D3,(SP)                ;DID TRAP OCCUR ON TEST INST
10908 053034  001403                    BEQ     1$                      ;YES GO ON
10909 053036  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10910 053042  104003                    ERROR   .+3             ;FPP ERROR
10911                                                           ;NO GO TO ERROR
10912 053044  022626            1$:     CMP     (SP)+,(SP)+             ;CLEAN UP STACK
10913 053046  170201                    STFPS   R1                      ;STORE FPS
10914 053050  022701  100000            CMP     #100000,R1              ;IS FPS CORRECT
10915 053054  001403                    BEQ     2$                      ;YES GO ON
10916 053056  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10917 053062  104003                    ERROR   .+3             ;FPP ERROR
10918                                                           ;NO GO TO ERROR
10919 053064  005004            2$:     CLR     R4                      ;INT R4 TO A KNOWN STATE
10920 053066  170304                    STST    R4                      ;STORE FEC AT R4
10921                                                           ;IF THE DESTINATION MODE IS IMPROPERLY
10922                                                           ;DECODED AN ODD ADDRESS TRAP TO 4
10923                                                           ;SHOULD OCCUR
10924 053070  022704  000002            CMP     #2,R4                   ;IS FEC CORRECT
10925 053074  001002                    BNE     3$                      ;NO GO TO ERROR
10926 053076  000167  177656            JMP     D4                      ;YES GO ON
10927 053102  004737  140132    3$:     CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10928 053106  104003                    ERROR   .+3             ;FPP ERROR
10929                                                           ;GO TO ERROR
10930 053110  000167  177644            JMP     D4                      ;THEN GO ON
10931                           ;
10932 053114  004737  140132    TIMEOU: CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10933 053120  104003                    ERROR   .+3             ;FPP ERROR
10934                                                           ;ERROR BECAUSE OF TRAP TO 4
10935 053122  000006                    RTT                             ;RETURN
10936                           ;
10937 053124  004737  140132    ILLOP2: CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10938 053130  104003                    ERROR   .+3             ;FPP ERROR
```

FLOATING POINT TESTS

```
10939
10940 053132  000006                          RTT                     ;ERROR BECAUSE OF TRAP TO 10
10941 053134                            FIN4:                         ;RETURN
10942                                   ;
10945 053134                            TSFP5:
10946                                   ;       TEST FID (INTERRUPT DISABLE BIT)
10947 053134  005037  140054                    CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10948 053140  013746  000244                    MOV     @#244,-(SP)             ;SAVE FP VECTOR
10949 053144  012737  053230  000244            MOV     #ILL,@#244              ;SETUP NEW VECTOR
10950 053152  012703  040000                    MOV     #40000,R3               ;SETUP DATA TO BE LOADED
10951 053156  170103                            LDFPS   R3                      ;LOAD FPS, FID=1
10952 053160  170020                            .WORD   170020                  ;ILLEGAL FP INSTRUCTION
10953 053162  170000                            CFCC
10954 053164  170201                            STFPS   R1                      ;SEE IF ERROR WAS RECORDED IN FPS
10955 053166  022701  140000                    CMP     #140000,R1              ;
10956 053172  001403                            BEQ     1$                      ;YES GO ON
10957 053174  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10958 053200  104003                            ERROR   +3                      ;FPP ERROR
10959                                                                           ;NO GO TO ERROR
10960 053202  170304                    1$:     STST    R4                      ;SEE IF FEC=2
10961 053204  022704  000002                    CMP     #2,R4                   ;
10962 053210  001403                            BEQ     2$                      ;YES GO ON
10963 053212  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10964 053216  104003                            ERROR   +3                      ;FPP ERROR
10965                                                                           ;NO GO TO ERROR
10966 053220  012637  000244            2$:     MOV     (SP)+,@#244             ;RESTORE VECTOR
10967                                   ;
10968 053224  000167  000010                    JMP     FIN5
10969                                   ;
10970 053230  004737  140132            ILL:    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10971 053234  104003                            ERROR   +3                      ;FPP ERROR
10972                                                                           ;FID ERROR
10973 053236  000006                            RTT                             ;RETURN
10974 053240                            FIN5:
10975                                   ;
10978 053240                            TSFP6:
10979                                   ;       TEST LDD, STD FSRC AND FDST MODE 1
10980 053240  005037  140054                    CLR     @#TRPFLG                ;CLEAR TRAP FLAG
10981 053244  005004                            CLR     R4                      ;SETUP TO LOAD DATA
10982 053246  170104                            LDFPS   R4                      ;CLEAR FPS
10983 053250  170011                            SETD                            ;SET FD TO 1
10984 053252  013746  000004                    MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
10985 053256  012737  053430  000004            MOV     #TSF6,@#4               ;SETUP NEW VECTOR
10986 053264  012704  053420                    MOV     #TS6DAT,R4              ;SETUP POINTER TO DATA
10987 053270  172414                            LDD     (R4),AC0                ; TEST INSTRUCTION
10988 053272  020427  053420                    CMP     R4,#TS6DAT              ;IS R4 CORRECT
10989 053276  001403                            BEQ     1$                      ;YES GO ON
10990 053300  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10991 053304  104003                            ERROR   +3                      ;FPP ERROR
10992                                                                           ;NO GO TO ERROR
10993 053306  012701  053410            1$:     MOV     #TS6DA,R1               ;SETUP POINTER TO DATA
10994 053312  012703  000004                    MOV     #4,R3                   ;INIT COUNTER
10995 053316  022421                    2$:     CMP     (R4)+,(R1)+             ;WAS SOURCE DATA ALTERED
10996 053320  001403                            BEQ     3$                      ;NO GO ON
10997 053322  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
10998 053326  104003                            ERROR   +3                      ;FPP ERROR
10999                                                                           ;YES GO TO ERROR
```

FLOATING POINT TESTS

```
11000 053330  077306          3$:     SOB     R3,2$                   ;ARE WE DONE
11001 053332  012704  003162            MOV     #TSTLOC,R4              ;SETUP POINTER FOR DATA
11002 053336  174014                    STD     AC0,(R4)                ; TEST INSTRUCTION
11003 053340  020427  003162            CMP     R4,#TSTLOC                      ;IS R4 CORRECT
11004 053344  001403                    BEQ     4$                      ;YES GO ON
11005 053346  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11006 053352  104003                    ERROR   .+3             ;FPP ERROR
11007                                                           ;NO GO TO ERROR
11008 053354  012701  053410      4$:   MOV     #TS6DA,R1               ;SETUP POINTER TO DATA
11009 053360  012703  000004            MOV     #4,R3                   ;INIT COUNTER
11010 053364  022421              5$:   CMP     (R4)+,(R1)+             ;IS DESTINATION DATA CORRECT
11011 053366  001403                    BEQ     6$                      ;YES GO ON
11012 053370  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11013 053374  104003                    ERROR   .+3             ;FPP ERROR
11014                                                           ;NO GO TO ERROR
11015 053376  077306          6$:     SOB     R3,5$                   ;ARE WE DONE
11016 053400  012637  000004            MOV     (SP)+,@#4               ;RESTORE VECTOR
11017
11018                         ;
11019 053404  000167  000030            JMP     FIN6
11020                         ;
11021 053410  177777          TS6DA:  .WORD   177777
11022 053412  000000                  .WORD   000000
11023 053414  052525                  .WORD   052525
11024 053416  125252                  .WORD   125252
11025 053420  177777          TS6DAT: .WORD   177777
11026 053422  000000                  .WORD   000000
11027 053424  052525                  .WORD   052525
11028 053426  125252                  .WORD   125252
11029                         ;
11030 053430  004737  140132    TSF6:   CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11031 053434  104003                    ERROR   .+3             ;FPP ERROR
11032                                                           ;ODD ADDRESS TRAP
11033 053436  000006                    RTT                             ;RETURN
11034 053440                    FIN6:
11035                         ;
11038 053440                    TSFP7:
11039                         ;       TEST LDD, LDF FSRC MODE 0
11040 053440  005037  140054            CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11041 053444  012704  000200            MOV     #200,R4                 ;SETUP TO LOAD FPS
11042 053450  170104                    LDFPS   R4                      ;LOAD FPS, FD=1
11043 053452  013746  000004            MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11044 053456  012737  053624  000004    MOV     #TSF7,@#4               ;SETUP NEW VECTOR
11045 053464  012704  053634            MOV     #TS7DA1,R4              ;SETUP POINTER TO DATA
11046 053470  172414                    LDD     (R4),AC0                ;CLEAR AC0
11047 053472  012701  053644            MOV     #TS7DA2,R1              ;SETUP POINTER TO DATA
11048 053476  172511                    LDD     (R1),AC1                ;LOAD AC1 WITH DATA
11049 053500  172401                    LDD     AC1,AC0                 ; TEST INSTRUCTION
11050 053502  012704  003162            MOV     #TSTLOC,R4              ;
11051 053506  174114                    STD     AC1,(R4)                ;CHECK IF AC1 HAS BEEN ALTERED
11052 053510  004767  000072            JSR     PC,CHECK7               ;
11053 053514  012704  003162            MOV     #TSTLOC,R4                      ;SETUP POINTERS FOR DATA
11054 053520  012701  053644            MOV     #TS7DA2,R1              ;
11055 053524  174014                    STD     AC0,(R4)                ;CHECK IF AC0 RECEIVED CORRECT DATA
11056 053526  004767  000054            JSR     PC,CHECK7               ;
11057 053532  012701  053634            MOV     #TS7DA1,R1              ;SETUP POINTER TO DATA
11058 053536  172511                    LDD     (R1),AC1                ;CLEAR AC1
```

FLOATING POINT TESTS

```
11059 053540  170001                        SETF                            ;SET FD=0
11060 053542  172401                        LDF     AC1,AC0                 ; TEST INSTRUCTION
11061 053544  170011                        SETD                            ;SET FD=1
11062 053546  012704  003162                MOV     #TSTLOC,R4              ;SETUP POINTER TO DATA
11063 053552  174114                        STD     AC1,(R4)                ;CHECK IF AC1 HAS BEEN ALTERED
11064 053554  004767  000026                JSR     PC,CHECK7               ;
11065 053560  012704  053654                MOV     #TS7DA4,R4              ;SETUP POINTERS FOR DATA
11066 053564  012701  003162                MOV     #TSTLOC,R1              ;
11067 053570  174011                        STD     AC0,(R1)                ;CHECK IF AC0 HAS CORRECT DATA
11068 053572  004767  000010                JSR     PC,CHECK7               ;
11069 053576  012637  000004                MOV     (SP)+,@#4               ;RESTORE VECTOR
11070
11071                               ;
11072 053602  000167  000056                JMP     FIN7
11073                               ;
11074 053606  012703  000004        CHECK7: MOV     #4,R3                   ;INIT COUNTER
11075 053612  022421                CHEK7:  CMP     (R4)+,(R1)+             ;IS DATA OK
11076 053614  001401                        BEQ     CHK7                    ;YES GO ON
11077 053616  104003                        ERROR   +3              ;FPP ERROR
11078                                                               ;NO GO TO ERROR
11079 053620  077304                CHK7:   SOB     R3,CHEK7                ;ARE WE DONE
11080 053622  000207                        RTS     PC                      ;YES RETURN
11081                               ;
11082 053624  004737  140132        TSF7:   CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11083 053630  104003                        ERROR   +3              ;FPP ERROR
11084                                                               ;ODD ADDRESS TRAP
11085 053632  000006                        RTT                             ;RETURN
11086                               ;
11087 053634  000000                TS7DA1: .WORD   0
11088 053636  000000                        .WORD   0
11089 053640  000000                        .WORD   0
11090 053642  000000                        .WORD   0
11091 053644  037641                TS7DA2: .WORD   37641
11092 053646  065121                        .WORD   65121
11093 053650  037373                        .WORD   37373
11094 053652  022265                        .WORD   22265
11095 053654  000000                TS7DA4: .WORD   0
11096 053656  000000                        .WORD   0
11097 053660  037373                        .WORD   37373
11098 053662  022265                        .WORD   22265
11099 053664                        FIN7:
11100                               ;
11103 053664                        TSFP10:
11104                               ;       TEST STD, STF FDST MODE 0
11105 053664  005037  140054                CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11106 053670  012704  000200                MOV     #200,R4                 ;SETUP TO LOAD FPS
11107 053674  170104                        LDFPS   R4                      ;LOAD FPS, FD=1
11108 053676  013746  000004                MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11109 053702  012737  054054  000004        MOV     #TSF10,@#4              ;SETUP NEW VECTOR
11110 053710  012704  054064                MOV     #TS10D1,R4              ;SETUP POINTER TO DATA
11111 053714  172414                        LDD     (R4),AC0                ;CLEAR AC0
11112 053716  012701  054074                MOV     #TS10D2,R1              ;SETUP POINTER TO DATA
11113 053722  172511                        LDD     (R1),AC1                ;LOAD AC1 WITH DATA
11114 053724  174100                        STD     AC1,AC0                 ; TEST INSTRUCTION
11115 053726  012704  003162                MOV     #TSTLOC,R4              ;
11116 053732  174114                        STD     AC1,(R4)                ;CHECK IF AC1 HAS BEEN ALTERED
11117 053734  004767  000072                JSR     PC,CHEC10               ;
```

FLOATING POINT TESTS

```
11118 053740  012704  003162                MOV     #TSTLOC,R4          ;SETUP POINTERS FOR DATA
11119 053744  012701  054074                MOV     #TS10D2,R1          ;
11120 053750  174014                        STD     AC0,(R4)            ;CHECK IF AC0 RECEIVED CORRECT DATA
11121 053752  004767  000054                JSR     PC,CHEC10           ;
11122 053756  012701  054064                MOV     #TS10D1,R1          ;SETUP POINTER TO DATA
11123 053762  172511                        LDD     (R1),AC1            ;CLEAR AC1
11124 053764  170001                        SETF                        ;SET FD=0
11125 053766  174100                        STF     AC1,AC0             ; TEST INSTRUCTION
11126 053770  170011                        SETD                        ;SET FD=1
11127 053772  012704  003162                MOV     #TSTLOC,R4          ;SETUP POINTER TO DATA
11128 053776  174114                        STD     AC1,(R4)            ;CHECK IF AC1 HAS BEEN ALTERED
11129 054000  004767  000026                JSR     PC,CHEC10           ;
11130 054004  012704  054104                MOV     #TS10D4,R4          ;SETUP POINTERS FOR DATA
11131 054010  012701  003162                MOV     #TSTLOC,R1          ;
11132 054014  174011                        STD     AC0,(R1)            ;CHECK IF AC0 HAS CORRECT DATA
11133 054016  004767  000010                JSR     PC,CHEC10           ;
11134 054022  012637  000004                MOV     (SP)+,@#4           ;RESTORE VECTOR
11135
11136                               ;
11137 054026  000167  000062                JMP     FIN10
11138                               ;
11139 054032  012703  000004        CHEC10: MOV     #4,R3               ;INIT COUNTER
11140 054036  022421                CH10:   CMP     (R4)+,(R1)+         ;IS DATA OK
11141 054040  001403                        BEQ     CHK10               ;YES GO ON
11142 054042  004737  140132                CALL    @#DETFPA            ;DETERMINE FLOATING POINT FAULT. $$$
11143 054046  104003                        ERROR   +3                  ;FPP ERROR
11144                                                                     ;NO GO TO ERROR
11145 054050  077306                CHK10:  SOB     R3,CH10             ;ARE WE DONE
11146 054052  000207                        RTS     PC                  ;YES RETURN
11147                               ;
11148 054054  004737  140132        TSF10:  CALL    @#DETFPA            ;DETERMINE FLOATING POINT FAULT. $$$
11149 054060  104003                        ERROR   +3                  ;FPP ERROR
11150                                                                     ;ODD ADDRESS TRAP
11151 054062  000006                        RTT                         ;
11152                               ;
11153 054064  000000                TS10D1: .WORD   0
11154 054066  000000                        .WORD   0
11155 054070  000000                        .WORD   0
11156 054072  000000                        .WORD   0
11157 054074  177777                TS10D2: .WORD   177777
11158 054076  111236                        .WORD   111236
11159 054100  100045                        .WORD   100045
11160 054102  003651                        .WORD   3651
11161 054104  000000                TS10D4: .WORD   0
11162 054106  000000                        .WORD   0
11163 054110  100045                        .WORD   100045
11164 054112  003651                        .WORD   3651
11165 054114                        FIN10:
11166                               ;
11169 054114                        TSFP11:
11170                               ;       TEST FDST SINGLE OPERAND MODE 0
11171 054114  005037  140054                CLR     @#TRPFLG            ;CLEAR TRAP FLAG
11172 054120  012704  000200                MOV     #200,R4             ;SETUP TO LOAD FPS
11173 054124  170104                        LDFPS   R4                  ;SET FD=1
11174 054126  012704  054210                MOV     #TS11D1,R4          ;SETUP POINTER TO DATA
11175 054132  172414                        LDD     (R4),AC0            ;LOAD ALL ONES TO AC0
11176 054134  170400                        CLRD    AC0                 ; TEST INSTRUCTION
```

FLOATING POINT TESTS

```
11177 054136  170203                    STFPS   R3                      ;GET FPS
11178 054140  012704  003162              MOV     #TSTLOC,R4              ;
11179 054144  174014                    STD     AC0,(R4)                ;CHECK AC0 FOR ALL ZEROES
11180 054146  012701  000004              MOV     #4,R1                   ;INIT COUNTER
11181 054152  022427  000000    1$:       CMP     (R4)+,#0                ;
11182 054156  001403                    BEQ     2$                      ;OK GO ON
11183 054160  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11184 054164  104003                    ERROR   +3              ;FPP ERROR
11185                                                           ;NO GO TO ERROR
11186 054166  077107            2$:     SOB     R1,1$                   ;ARE WE DONE
11187 054170  020327  000204              CMP     R3,#204                 ;CHECK FPS
11188 054174  001403                    BEQ     3$                      ;OK GO ON
11189 054176  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11190 054202  104003                    ERROR   +3              ;FPP ERROR
11191                                                           ;NO GO TO ERROR
11192 054204                    3$:
11193                           ;
11194 054204  000167  000010              JMP     FIN11
11195                           ;
11196 054210  177777            TS11D1: .WORD   177777
11197 054212  177777                    .WORD   177777
11198 054214  177777                    .WORD   177777
11199 054216  177777                    .WORD   177777
11200 054220                    FIN11:
11201                           ;
11204 054220                    TSFP12:
11205                           ;       TEST FDST SOP MODE 0 WITH ILLEGAL AC7
11206 054220  005037  140054              CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11207 054224  012703  040200              MOV     #40200,R3               ;SETUP TO LOAD FPS
11208 054230  170103                    LDFPS   R3                      ;SET FID=1, AND FD=1
11209 054232  170407                    CLRD    AC7                     ; TEST INSTRUCTION
11210 054234  170204                    STFPS   R4                      ;GET FPS
11211 054236  170305                    STST    R5                      ;GET FEC
11212 054240  022704  140200              CMP     #140200,R4              ;IS FPS CORRECT
11213 054244  001403                    BEQ     1$                      ;YES GO ON
11214 054246  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11215 054252  104003                    ERROR   +3              ;FPP ERROR
11216                                                           ;NO GO TO ERROR
11217 054254  022705  000002    1$:     CMP     #2,R5                   ;IS FEC CORRECT
11218 054260  001403                    BEQ     2$                      ;YES GO ON
11219 054262  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11220 054266  104003                    ERROR   +3              ;FPP ERROR
11221                                                           ;NO GO TO ERROR
11222 054270                    2$:
11223                           ;
11226 054270                    TSFP13:
11227                           ;       TEST FDST SOP MODE 1
11228 054270  013746  000004              MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11229 054274  012737  054424  000004      MOV     #TSF13,@#4              ;SETUP NEW VECTOR
11230 054302  005037  140054              CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11231 054306  012702  000200              MOV     #2(  ,R2                ;SETUP TO LOAD FPS
11232 054312  170102                    LDFPS   R2                      ;SET FD=1
11233 054314  012705  000004              MOV     #4,R5                   ;INIT COUNTER
11234 054320  012704  003162              MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
11235 054324  012724  177777    100$:   MOV     #177777,(R4)+           ;MOVE ALL ONES TO TEST LOCATION
11236 054330  077503                    SOB     R5,100$                 ;ARE WE DONE
11237 054332  012702  003162              MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
```

FLOATING POINT TESTS

```
11238 054336  170412                          CLRD    (R2)            ; TEST INSTRUCTION
11239 054340  170203                          STFPS   R3              ;GET FPS
11240 054342  020227  003162                  CMP     R2,#TSTLOC              ;WAS R2 ALTERED
11241 054346  001403                          BEQ     1$              ;NO GO ON
11242 054350  004737  140132                  CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11243 054354  104003                          ERROR   +3      ;FPP ERROR
11244                                         ;YES GO TO ERROR
11245 054356  012701  000004          1$:     MOV     #4,R1           ;INIT COUNTER
11246 054362  022227  000000          2$:     CMP     (R2)+,#0        ;CHECK LOCATION FOR 0
11247 054366  001403                          BEQ     3$              ;OK GO ON
11248 054370  004737  140132                  CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11249 054374  104003                          ERROR   +3      ;FPP ERROR
11250                                         ;NO GO TO ERROR
11251 054376  077107                  3$:     SOB     R1,2$           ;ARE WE DONE
11252 054400  020327  000204                  CMP     R3,#204         ;CHECK FPS
11253 054404  001403                          BEQ     4$              ;OK GO ON
11254 054406  004737  140132                  CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11255 054412  104003                          ERROR   +3      ;FPP ERROR
11256                                         ;NO GO TO ERROR
11257 054414  012637  000004          4$:     MOV     (SP)+,@#4       ;RESTORE VECTOR
11258
11259                                 ;
11260 054420  000167  000010                  JMP     FIN13
11261                                 ;
11262 054424  004737  140132          TSF13:  CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11263 054430  104003                          ERROR   +3      ;FPP ERROR
11264                                         ;ODD ADDRESS TRAP
11265 054432  000006                          RTT             ;RETURN
11266                                 ;
11267 054434                          FIN13:
11268                                 ;
11271 054434                          TSFP14:
11272                                 ;       TEST FDST SOP MODE 2
11273 054434  013746  000004                  MOV     @#4,-(SP)       ;SAVE TIMEOUT VECTOR
11274 054440  012737  054574  000004          MOV     #TSF14,@#4      ;SETUP NEW VECTOR
11275 054446  005037  140054                  CLR     @#TRPFLG        ;CLEAR TRAP FLAG
11276 054452  012702  000200                  MOV     #200,R2         ;SETUP TO LOAD FPS
11277 054456  170102                          LDFPS   R2              ;SET FD=1
11278 054460  012705  000004                  MOV     #4,R5           ;INIT COUNTER
11279 054464  012704  003162                  MOV     #TSTLOC,R4      ;SETUP POINTER TO TEST LOCATION
11280 054470  012724  177777          100$:   MOV     #177777,(R4)+   ;MOVE ALL ONES TO TEST LOCATION
11281 054474  077503                          SOB     R5,100$         ;ARE WE DONE
11282 054476  012702  003162                  MOV     #TSTLOC,R2              ;SETUP POINTER TO DATA
11283 054502  170422                          CLRD    (R2)+           ; TEST INSTRUCTION
11284 054504  170203                          STFPS   R3              ;GET FPS
11285 054506  020227  003172                  CMP     R2,#TSTLOC+10           ;IS R2 CORRECT
11286 054512  001403                          BEQ     1$              ;YES GO ON
11287 054514  004737  140132                  CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11288 054520  104003                          ERROR   +3      ;FPP ERROR
11289                                         ;NO GO TO ERROR
11290 054522  012702  003162          1$:     MOV     #TSTLOC,R2              ;SETUP POINTER TO DATA
11291 054526  012701  000004                  MOV     #4,R1           ;INIT COUNTER
11292 054532  022227  000000          2$:     CMP     (R2)+,#0        ;CHECK LOCATION FOR 0
11293 054536  001403                          BEQ     3$              ;YES GO ON
11294 054540  004737  140132                  CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11295 054544  104003                          ERROR   +3      ;FPP ERROR
11296                                         ;NO GO TO ERROR
```

FLOATING POINT TESTS

```
11297 054546  077107                3$:     SOB     R1,2$                   ;ARE WE DONE
11298 054550  020327  000204                CMP     R3,#204                 ;CHECK FPS
11299 054554  001403                        BEQ     4$                      ;OK GO ON
11300 054556  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11301 054562  104003                        ERROR   +3              ;FPP ERROR
11302                                                                 ;NO GO TO ERROR
11303 054564  012637  000004          4$:     MOV     (SP)+,@#4               ;RESTORE VECTOR
11304
11305                                       ;
11306 054570  000167  000010                JMP     FIN14
11307                                       ;
11308 054574  004737  140132          TSF14:  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11309 054600  104003                        ERROR   +3              ;FPP ERROR
11310                                                                 ;ODD ADDRESS TRAP
11311 054602  000006                        RTT                             ;RETURN
11312                                       ;
11313 054604  000240                FIN14:  NOP
11314                                       ;
11317                                       ;
11318                                       ;---------------------------------
11319                                       ;TEST  FDST SOP MODE 3
11320                                       ;
11321                                       ;
11322 054606                        TSFP15:
11323                                       ;
11324 054606  013746  000004                MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11325 054612  012737  055012  000004        MOV     #TSF15,@#4              ;SETUP NEW VECTOR
11326 054620  005037  140054                CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11327 054624  012702  000200                MOV     #200,R2                 ;SETUP TO LOAD FPS
11328 054630  170102                        LDFPS   R2                      ;SET FD=1
11329 054632  012705  000011                MOV     #9.,R5                  ;INIT COUNTER
11330 054636  012704  003162                MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
11331 054642  012724  177777          100$:   MOV     #177777,(R4)+           ;INIT TEST LOCATION
11332 054646  077503                        SOB     R5,100$                 ;ARE WE DONE
11333 054650  012737  003174  003162        MOV     #TSTLOC+12,@#TSTLOC             ;INIT TEST LOCATION
11334 054656  012702  003162                MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11335 054662  170432                        CLRD    @(R2)+                  ; TEST INSTRUCTION
11336 054664  170203                        STFPS   R3                      ;GET FPS
11337 054666  020227  003164                CMP     R2,#TSTLOC+2                    ;IS R2 CORRECT
11338 054672  001403                        BEQ     1$                      ;YES GO ON
11339 054674  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11340 054700  104003                        ERROR   +3              ;FPP ERROR
11341                                                                 ;NO GO TO ERROR
11342 054702  012702  003162          1$:     MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11343 054706  022227  003174                CMP     (R2)+,#TSTLOC+12                ;IS DATA CORRECT
11344 054712  001403                        BEQ     2$                      ;YES GO ON
11345 054714  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11346 054720  104003                        ERROR   +3              ;FPP ERROR
11347                                                                 ;NO GO TO ERROR
11348 054722  012701  000004          2$:     MOV     #4,R1                   ;INIT COUNTER
11349 054726  022227  177777          3$:     CMP     (R2)+,#177777           ;IS LOCATION ALL ONES
11350 054732  001403                        BEQ     4$                      ;YES GO ON
11351 054734  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11352 054740  104003                        ERROR   +3              ;FPP ERROR
11353                                                                 ;NO GO TO ERROR
11354
11355 054742  077107                4$:     SOB     R1,3$                   ;ARE WE DONE
```

FLOATING POINT TESTS

```
11356 054744  012701  000004            MOV     #4,R1           ;INIT COUNTER
11357 054750  022227  000000     5$:    CMP     (R2)+,#0        ;IS LOCATION 0
11358 054754  001403                     BEQ     6$              ;YES GO ON
11359 054756  004737  140132             CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11360 054762  104003                     ERROR   +3      ;FPP ERROR
11361                                                     ;NO GO TO ERROR
11362 054764  077107     6$:    SOB     R1,5$           ;ARE WE DONE
11363 054766  020327  000204             CMP     R3,#204         ;CHECK FPS
11364 054772  001403                     BEQ     7$              ;OK GO ON
11365 054774  004737  140132             CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11366 055000  104003                     ERROR   +3      ;FPP ERROR
11367                                                     ;NO GO TO ERROR
11368 055002  012637  000004     7$:    MOV     (SP)+,@#4       ;RESTORE VECTOR
11369 055006  000167  000010             JMP     FIN15
11370                            ;
11371 055012  004737  140132     TSF15: CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11372 055016  104003                     ERROR   +3      ;FPP ERROR
11373                                                     ;ODD ADDRESS TRAP
11374 055020  000006                     RTT             ;RETURN
11375                            ;
11376 055022  000240     FIN15: NOP
11377                            ;
11380                            ;
11381                            ;---------------------------------
11382                            ;TEST  FDST SOP MODE 4
11383                            ;
11384                            ;
11385 055024             TSFP16:
11386                            ;
11387 055024  013746  000004                MOV     @#4,-(SP)       ;SAVE TIMEOUT VECTOR
11388 055030  012737  055202  000004        MOV     #TSF16,@#4      ;SETUP NEW VECTOR
11389 055036  005037  140054                CLR     @#TRPFLG        ;CLEAR TRAP FLAG
11390 055042  012702  000200                MOV     #200,R2         ;SETUP TO LOAD FPS
11391 055046  170102                        LDFPS   R2              ;SET FD=1
11392 055050  012705  000010                MOV     #8.,R5          ;INIT COUNTER
11393 055054  012704  003162                MOV     #TSTLOC,R4      ;SETUP POINTER TO TEST LOCATION
11394 055060  012724  177777     100$:      MOV     #177777,(R4)+   ;INIT TEST LOCATION
11395 055064  077503                        SOB     R5,100$         ;ARE WE DONE
11396 055066  012702  003172                MOV     #TSTLOC+10,R2           ;SETUP POINTER TO DATA
11397 055072  170442                        CLRD    -(R2)           ; TEST INSTRUCTION
11398 055074  170203                        STFPS   R3              ;GET FPS
11399 055076  020227  003162                CMP     R2,#TSTLOC              ;IS R2 CORRECT
11400 055102  001403                        BEQ     1$              ;YES GO ON
11401 055104  004737  140132                CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11402 055110  104003                        ERROR   +3      ;FPP ERROR
11403                                                   ;NO GO TO ERROR
11404 055112  012701  000004     1$:    MOV     #4,R1           ;INIT COUNTER
11405 055116  022227  000000     2$:    CMP     (R2)+,#0        ;IS LOCATION 0
11406 055122  001403                     BEQ     3$              ;YES GO ON
11407 055124  004737  140132             CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
11408 055130  104003                     ERROR   +3      ;FPP ERROR
11409                                                     ;NO GO TO ERROR
11410 055132  077107     3$:    SOB     R1,2$           ;ARE WE DONE
11411 055134  012701  000004             MOV     #4,R1           ;INIT COUNTER
11412 055140  022227  177777     4$:    CMP     (R2)+,#177777   ;IS LOCATION UNCHANGED
11413 055144  001403                     BEQ     5$              ;YES GO ON
11414 055146  004737  140132             CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
```

FLOATING POINT TESTS

```
11415 055152  104003                    ERROR   +3              ;FPP ERROR
11416                                                           ;NO GO TO ERROR
11417 055154  077107              5$:   SOB     R1,4$                   ;ARE WE DONE
11418 055156  020327  000204            CMP     R3,#204                 ;CHECK FPS
11419 055162  001403                    BEQ     6$                      ;OK GO ON
11420 055164  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11421 055170  104003                    ERROR   +3              ;FPP ERROR
11422                                                           ;NO GO TO ERROR
11423 055172  012637  000004      6$:   MOV     (SP)+,@#4               ;RESTORE VECTOR
11424 055176  000167  000010            JMP     FIN16
11425                                   ;
11426 055202  004737  140132      TSF16: CALL   @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11427 055206  104003                    ERROR   +3              ;FPP ERROR
11428                                                           ;ODD ADDRESS TRAP
11429 055210  000006                    RTT                             ;RETURN
11430                                   ;
11431 055212  000240      FIN16: NOP
11432                                   ;
11435                                   ;
11436                                   ;---------------------------------
11437                                   ;TEST  FDST SOP MODE 5
11438                                   ;
11439                                   ;
11440 055214              TSFP17:
11441                                   ;
11442 055214  013746  000004            MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11443 055220  012737  055414  000004    MOV     #TSF17,@#4              ;SETUP NEW VECTOR
11444 055226  005037  140054            CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11445 055232  012702  000200            MOV     #200,R2                 ;SETUP TO LOAD FPS
11446 055236  170102                    LDFPS   R2                      ;SET FD=1
11447 055240  012705  000011            MOV     #9.,R5                  ;INIT COUNTER
11448 055244  012704  003162            MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
11449 055250  012724  177777      100$: MOV     #177777,(R4)+           ;INIT TEST LOCATION
11450 055254  077503                    SOB     R5,100$                 ;ARE WE DONE
11451 055256  012737  003174  003162    MOV     #TSTLOC+12,@#TSTLOC             ;INIT TEST LOCATION
11452 055264  012702  003164            MOV     #TSTLOC+2,R2                    ;SETUP POINTER TO DATA
11453 055270  170452                    CLRD    @-(R2)                  ; TEST INSTRUCTION
11454 055272  170203                    STFPS   R3                      ;GET FPS
11455 055274  020227  003162            CMP     R2,#TSTLOC                      ;IS R2 CORRECT
11456 055300  001403                    BEQ     1$                      ;YES GO ON
11457 055302  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11458 055306  104003                    ERROR   +3              ;FPP ERROR
11459                                                           ;NO GO TO ERROR
11460 055310  022227  003174      1$:   CMP     (R2)+,#TSTLOC+12                ;IS DATA CORRECT
11461 055314  001403                    BEQ     2$                      ;YES GO ON
11462 055316  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11463 055322  104003                    ERROR   +3              ;FPP ERROR
11464                                                           ;NO GO TO ERROR
11465 055324  012701  000004      2$:   MOV     #4,R1                   ;INIT COUNTER
11466 055330  022227  177777      3$:   CMP     (R2)+,#177777           ;IS LOCATION ALL ONES
11467 055334  001403                    BEQ     4$                      ;YES GO ON
11468 055336  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11469 055342  104003                    ERROR   +3              ;FPP ERROR
11470                                                           ;NO GO TO ERROR
11471 055344  077107              4$:   SOB     R1,3$                   ;ARE WE DONE
11472 055346  012701  000004            MOV     #4,R1                   ;INIT COUNTER
11473 055352  022227  000000      5$:   CMP     (R2)+,#0                ;IS LOCATION 0
```

FLOATING POINT TESTS

```
11474 055356  001403                    BEQ     6$                      ;YES GO ON
11475 055360  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11476 055364  104003                    ERROR   +3              ;FPP ERROR
11477                                                           ;NO GO TO ERROR
11478 055366  077107            6$:     SOB     R1,5$                   ;ARE WE DONE
11479 055370  020327  000204            CMP     R3,#204                 ;CHECK FPS
11480 055374  001403                    BEQ     7$                      ;OK GO ON
11481 055376  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11482 055402  104003                    ERROR   +3              ;FPP ERROR
11483                                                           ;NO GO TO ERROR
11484 055404  012637  000004    7$:     MOV     (SP)+,@#4               ;RESTORE VECTOR
11485 055410  000167  000010            JMP     FIN17
11486                                   ;
11487 055414  004737  140132    TSF17:  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11488 055420  104003                    ERROR   +3              ;FPP ERROR
11489                                                           ;ODD ADDRESS TRAP
11490 055422  000006                    RTT                             ;RETURN
11491                                   ;
11492 055424  000240    FIN17:  NOP
11493                                   ;
11496                                   ;
11497                                   ;---------------------------------
11498                                   ;TEST  FDST SOP MODE 6
11499                                   ;
11500                                   ;
11501 055426            TSFP20:
11502                                   ;
11503 055426  005037  140054            CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11504 055432  013746  000004            MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11505 055436  012737  055612  000004    MOV     #TSF20,@#4              ;SETUP NEW VECTOR
11506 055444  012702  000200            MOV     #200,R2                 ;SETUP TO LOAD FPS
11507 055450  170102                    LDFPS   R2                      ;SET FD=1
11508 055452  012705  000010            MOV     #8.,R5                  ;INIT COUNTER
11509 055456  012704  003162            MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
11510 055462  012724  177777    100$:   MOV     #177777,(R4)+           ;INIT TEST LOCATION
11511 055466  077503                    SOB     R5,100$                 ;ARE WE DONE
11512 055470  012702  003163            MOV     #TSTLOC+1,R2                    ;SETUP POINTER TO DATA
11513 055474  170462  000007            CLRD    7(R2)                   ; TEST INSTRUCTION
11514 055500  170203                    STFPS   R3                      ;GET FPS
11515 055502  020227  003163            CMP     R2,#TSTLOC+1                    ;IS R2 CORRECT
11516 055506  001403                    BEQ     1$                      ;YES GO ON
11517 055510  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11518 055514  104003                    ERROR   +3              ;FPP ERROR
11519                                                           ;NO GO TO ERROR
11520 055516  012702  003162    1$:     MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11521 055522  012701  000004            MOV     #4,R1                   ;INIT COUNTER
11522 055526  022227  177777    2$:     CMP     (R2)+,#177777           ;IS DATA CORRECT
11523 055532  001403                    BEQ     3$                      ;YES GO ON
11524 055534  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11525 055540  104003                    ERROR   +3              ;FPP ERROR
11526                                                           ;NO GO TO ERROR
11527 055542  077107    3$:     SOB     R1,2$                   ;ARE WE DONE
11528 055544  012701  000004            MOV     #4,R1                   ;INIT COUNTER
11529 055550  022227  000000    4$:     CMP     (R2)+,#0                ;IS DATA CORRECT
11530 055554  001403                    BEQ     5$                      ;YES GO ON
11531 055556  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11532 055562  104003                    ERROR   +3              ;FPP ERROR
```

FLOATING POINT TESTS

```
11533                                                          ;NO GO TO ERROR
11534 055564  077107                  5$:     SOB     R1,4$            ;ARE WE DONE
11535 055566  020327  000204                  CMP     R3,#204          ;IS FPS CORRECT
11536 055572  001403                  BEQ     6$               ;YES GO ON
11537 055574  004737  140132          CALL    @#DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
11538 055600  104003                  ERROR   +3               ;FPP ERROR
11539                                                          ;NO GO TO ERROR
11540 055602  012637  000004          6$:     MOV     (SP)+,@#4        ;RESTORE VECTOR
11541 055606  000167  000010          JMP     FIN20
11542                                 ;
11543 055612  004737  140132          TSF20:  CALL    @#DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
11544 055616  104003                  ERROR   +3               ;FPP ERROR
11545                                                          ;ODD ADDRESS TRAP
11546 055620  000006                  RTT                      ;RETURN
11547                                 ;
11548 055622  000240                  FIN20:  NOP
11549                                 ;
11552                                 ;
11553                                 ;---------------------------------
11554                                 ;TEST  FDST SOP MODE 7
11555                                 ;
11556                                 ;
11557 055624                          TSFP21:
11558                                 ;
11559 055624  005037  140054                  CLR     @#TRPFLG                 ;CLEAR TRAP FLAG
11560 055630  013746  000004                  MOV     @#4,-(SP)                ;SAVE TIMEOUT VECTOR
11561 055634  012737  056032  000004          MOV     #TSF21,@#4               ;SETUP NEW VECTOR
11562 055642  012702  000200                  MOV     #200,R2                  ;SETUP TO LOAD FPS
11563 055646  170102                  LDFPS   R2                       ;SET FD=1
11564 055650  012705  000010                  MOV     #8.,R5                   ;INIT COUNTER
11565 055654  012704  003162                  MOV     #TSTLOC,R4                       ;SETUP POINTER TO TEST LOCATION
11566 055660  012724  177777          100$:   MOV     #177777,(R4)+    ;INIT TEST LOCATION
11567 055664  077503                  SOB     R5,100$                  ;ARE WE DONE
11568 055666  012737  003162  003172          MOV     #TSTLOC,@#TSTLOC+10              ;INIT TEST LOCATION
11569 055674  012702  003165                  MOV     #TSTLOC+3,R2             ;SETUP POINTER TO DATA
11570 055700  170472  000005                  CLRD    @5(R2)                   ; TEST INSTRUCTION
11571 055704  170203                  STFPS   R3                       ;GET FPS
11572 055706  020227  003165                  CMP     R2,#TSTLOC+3                     ;IS R2 CORRECT
11573 055712  001403                  BEQ     1$               ;YES GO ON
11574 055714  004737  140132          CALL    @#DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
11575 055720  104003                  ERROR   +3               ;FPP ERROR
11576                                                          ;NO GO TO ERROR
11577 055722  012702  003162          1$:     MOV     #TSTLOC,R2               ;SETUP POINTER TO DATA
11578 055726  012701  000004                  MOV     #4,R1                    ;INIT COUNTER
11579 055732  022227  000000          2$:     CMP     (R2)+,#0                 ;IS DATA CORRECT
11580 055736  001403                  BEQ     3$               ;YES GO ON
11581 055740  004737  140132          CALL    @#DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
11582 055744  104003                  ERROR   +3               ;FPP ERROR
11583                                                          ;NO GO TO ERROR
11584 055746  077107                  3$:     SOB     R1,2$            ;ARE WE DONE
11585 055750  022227  003162                  CMP     (R2)+,#TSTLOC            ;IS DATA CORRECT
11586 055754  001403                  BEQ     4$               ;YES GO ON
11587 055756  004737  140132          CALL    @#DETFPA         ;DETERMINE FLOATING POINT FAULT. $$$
11588 055762  104003                  ERROR   +3               ;FPP ERROR
11589                                                          ;NO GO TO ERROR
11590 055764  012701  000003          4$:     MOV     #3,R1            ;INIT COUNTER
11591 055770  022227  177777          5$:     CMP     (R2)+,#177777            ;IS DATA CORRECT
```

FLOATING POINT TESTS

```
11592 055774  001403                    BEQ     6$                      ;YES GO ON
11593 055776  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11594 056002  104003                    ERROR   +3                    ;FPP ERROR
11595                                                                 ;NO GO TO ERROR
11596 056004  077107            6$:     SOB     R1,5$                   ;ARE WE DONE
11597 056006  020327  000204              CMP     R3,#204                 ;CHECK FPS
11598 056012  001403                    BEQ     7$                      ;OK GO ON
11599 056014  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11600 056020  104003                    ERROR   +3                    ;FPP ERROR
11601                                                                 ;NO GO TO ERROR
11602 056022  012637  000004      7$:     MOV     (SP)+,@#4               ;RESTORE VECTOR
11603 056026  000167  000010              JMP     FIN21
11604                             ;
11605 056032  004737  140132      TSF21:  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11606 056036  104003                    ERROR   +3                    ;FPP ERROR
11607                                                                 ;ODD ADDRESS TRAP
11608 056040  000006                    RTT                             ;RETURN
11609                             ;
11610 056042  000240              FIN21:  NOP
11611                             ;
11614                             ;
11615                             ;---------------------------------
11616                             ;TEST  FDST SOP MODE 3 GR7
11617                             ;
11618                             ;
11619 056044                      TSFP22:
11620                             ;
11621 056044  005037  140054              CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11622 056050  013746  000004              MOV     @#4,-(SP)               ;SAVE TIME OUT VECTOR
11623 056054  012737  056216  000004      MOV     #TSF22,@#4              ;SETUP NEW VECTOR
11624 056062  012702  000200              MOV     #200,R2                 ;SETUP TO LOAD FPS
11625 056066  170102                    LDFPS   R2                      ;SET FD=1
11626 056070  012705  000010              MOV     #8.,R5                  ;INIT COUNTER
11627 056074  012704  003162              MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
11628 056100  012724  177777      100$:   MOV     #177777,(R4)+           ;INIT TEST LOCATION
11629 056104  077503                    SOB     R5,100$                 ;ARE WE DONE
11630 056106  012737  003172  003162      MOV     #TSTLOC+10,@#TSTLOC             ;INIT TEST LOCATION
11631 056114  170437  003162              CLRD    @#TSTLOC                        ; TEST INSTRUCTION
11632 056120  170203                    STFPS   R3                      ;GET FPS
11633 056122  012702  003162              MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11634 056126  012701  000004              MOV     #4,R1                   ;INIT COUNTER
11635 056132  022227  000000      1$:     CMP     (R2)+,#0                ;IS DATA CORRECT
11636 056136  001403                    BEQ     2$                      ;YES GO ON
11637 056140  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11638 056144  104003                    ERROR   +3                    ;FPP ERROR
11639                                                                 ;NO GO TO ERROR
11640 056146  077107            2$:     SOB     R1,1$                   ;ARE WE DONE
11641 056150  012701  000004              MOV     #4,R1                   ;INIT COUNTER
11642 056154  022227  177777      3$:     CMP     (R2)+,#177777           ;IS DATA CORRECT
11643 056160  001403                    BEQ     4$                      ;YES GO ON
11644 056162  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11645 056166  104003                    ERROR   +3                    ;FPP ERROR
11646                                                                 ;NO GO TO ERROR
11647 056170  077107            4$:     SOB     R1,3$                   ;ARE WE DONE
11648 056172  020327  000204              CMP     R3,#204                 ;CHECK FPS
11649 056176  001403                    BEQ     5$                      ;OK GO ON
11650 056200  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
```

FLOATING POINT TESTS

```
11651 056204  104003                    ERROR   ;3              ;FPP ERROR
11652                                                           ;NO GO TO ERROR
11653 056206  012637  000004      5$:     MOV     (SP)+,@#4               ;RESTORE VECTOR
11654 056212  000167  000010              JMP     FIN22
11655                             ;
11656 056216  004737  140132      TSF22:  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11657 056222  104003                    ERROR   ;3              ;FPP ERROR
11658                                                           ;ODD ADDRESS TRAP
11659 056224  000006                    RTT                             ;RETURN
11660                             ;
11661 056226  000240              FIN22:  NOP
11662                             ;
11665                             ;
11666                             ;---------------------------------
11667                             ;TEST  FDST SOP MODE 6 GR7
11668                             ;
11669                             ;
11670 056230                      TSFP23:
11671                             ;
11672 056230  005037  140054              CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11673 056234  013746  000004              MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11674 056240  012737  056352  000004      MOV     #TSF23,@#4              ;SETUP NEW VECTOR
11675 056246  012702  000200              MOV     #200,R2                 ;SETUP TO LOAD FPS
11676 056252  170102                      LDFPS   R2                      ;SET FD=1
11677 056254  012705  000004              MOV     #4,R5                   ;INIT COUNTER
11678 056260  012704  003162              MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
11679 056264  012724  177777      100$:   MOV     #177777,(R4)+           ;INIT TEST LOCATION
11680 056270  077503                      SOB     R5,100$                 ;ARE WE DONE
11681 056272  170467  124664              CLRD    TSTLOC                  ; TEST INSTRUCTION
11682 056276  170203                      STFPS   R3                      ;GET FPS
11683 056300  012701  000004              MOV     #4,R1                   ;INIT COUNTER
11684 056304  012702  003162              MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11685 056310  022227  000000      1$:     CMP     (R2)+,#0                ;IS DATA CORRECT
11686 056314  001403                      BEQ     2$                      ;YES GO ON
11687 056316  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11688 056322  104003                    ERROR   ;3              ;FPP ERROR
11689                                                           ;NO GO TO ERROR
11690 056324  077107              2$:     SOB     R1,1$                   ;ARE WE DONE
11691 056326  020327  000204              CMP     R3,#204                 ;CHECK FPS
11692 056332  001403                      BEQ     3$                      ;OK GO ON
11693 056334  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11694 056340  104003                    ERROR   ;3              ;FPP ERROR
11695                                                           ;NO GO TO ERROR
11696 056342  012637  000004      3$:     MOV     (SP)+,@#4               ;RESTORE VECTOR
11697 056346  000167  000010              JMP     FIN23
11698                             ;
11699 056352  004737  140132      TSF23:  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11700 056356  104003                    ERROR   ;3              ;FPP ERROR
11701                                                           ;ODD ADDRESS TRAP
11702 056360  000006                      RTT                             ;
11703 056362  000240              FIN23:  NOP
11704                             ;
11707                             ;
11708                             ;---------------------------------
11709                             ;TEST  FDST SOP MODE 7 GR7
11710                             ;
11711                             ;
```

FLOATING POINT TESTS

```
11712 056364                          TSFP24:
11713                                 ;
11714 056364  005037  140054                CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11715 056370  013746  000004                MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
11716 056374  012737  056552  000004        MOV     #TSF24,@#4              ;SETUP NEW VECTOR
11717 056402  012702  000200                MOV     #200,R2                 ;SETUP TO LOAD FPS
11718 056406  170102                        LDFPS   R2                      ;SET FD=1
11719 056410  012705  000010                MOV     #8.,R5          ;INIT COUNTER
11720 056414  012704  003162                MOV     #TSTLOC,R4              ;SETUP TEST LOCATION POINTER
11721 056420  012724  177777          100$: MOV     #177777,(R4)+           ;INIT TEST LOCATION
11722 056424  077503                        SOB     R5,100$                 ;ARE WE DONE
11723 056426  012737  003162  003172        MOV     #TSTLOC,@#TSTLOC+10             ;INIT TEST LOCATION
11724 056434  170477  124532                CLRD    @TSTLOC+10                      ; TEST INSTRUCTION
11725 056440  170203                        STFPS   R3                      ;GET FPS
11726 056442  012702  003162                MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11727 056446  012701  000004                MOV     #4,R1                   ;INIT COUNTER
11728 056452  022227  000000          1$:   CMP     (R2)+,#0                ;IS DATA CORRECT
11729 056456  001403                        BEQ     2$                      ;YES GO ON
11730 056460  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11731 056464  104003                        ERROR   +3              ;FPP ERROR
11732                                                                 ;NO GO TO ERROR
11733 056466  077107                  2$:   SOB     R1,1$                   ;ARE WE DONE
11734 056470  022227  003162                CMP     (R2)+,#TSTLOC           ;IS DATA CORRECT
11735 056474  001403                        BEQ     3$                      ;YES GO ON
11736 056476  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11737 056502  104003                        ERROR   +3              ;FPP ERROR
11738                                                                 ;NO GO TO ERROR
11739 056504  012701  000003          3$:   MOV     #3,R1                   ;INIT COUNTER
11740 056510  022227  177777          4$:   CMP     (R2)+,#177777           ;IS DATA CORRECT
11741 056514  001403                        BEQ     5$                      ;YES GO ON
11742 056516  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11743 056522  104003                        ERROR   +3              ;FPP ERROR
11744                                                                 ;NO GO TO ERROR
11745 056524  077107                  5$:   SOB     R1,4$                   ;ARE WE DONE
11746 056526  020327  000204                CMP     R3,#204                 ;CHECK FPS
11747 056532  001403                        BEQ     6$                      ;OK GO ON
11748 056534  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11749 056540  104003                        ERROR   +3              ;FPP ERROR
11750                                                                 ;NO GO TO ERROR
11751 056542  012637  000004          6$:   MOV     (SP)+,@#4               ;RESTORE VECTOR
11752 056546  000167  000010                JMP     FIN24
11753                                 ;
11754 056552  004737  140132          TSF24: CALL   @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11755 056556  104003                        ERROR   +3              ;FPP ERROR
11756                                                                 ;ODD ADDRESS TRAP
11757 056560  000006                        RTT                             ;
11758                                 ;
11759 056562  000240                FIN24:  NOP
11760                                 ;
11763                                 ;
11764                                 ;---------------------------------
11765                                 ;TEST  CLRF
11766                                 ;
11767                                 ;
11768 056564                          TSFP25:
11769                                 ;
11770 056564  005037  140054                CLR     @#TRPFLG                ;CLEAR TRAP FLAG
```

FLOATING POINT TESTS

```
11771 056570  005002                      CLR     R2                      ;SETUP TO LOAD FPS
11772 056572  170102                      LDFPS   R2                      ;SET FD=0
11773 056574  012705  000004              MOV     #4,R5                   ;INIT COUNTER
11774 056600  012704  003162              MOV     #TSTLOC,R4              ;SETUP POINTER TO TEST LOCATION
11775 056604  012724  177777      100$:   MOV     #177777,(R4)+           ;INIT TEST LOCATION
11776 056610  077503                      SOB     R5,100$                 ;ARE WE DONE
11777 056612  012702  003162              MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11778 056616  170422                      CLRF    (R2)+                   ; TEST INSTRUCTION
11779 056620  170203                      STFPS   R3                      ;GET FPS
11780 056622  020227  003166              CMP     R2,#TSTLOC+4                    ;IS R2 CORRECT
11781 056626  001403                      BEQ     1$                      ;YES GO ON
11782 056630  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11783 056634  104003                      ERROR   +3                      ;FPP ERROR
11784                                                                     ;NO GO TO ERROR
11785 056636  012702  003162      1$:     MOV     #TSTLOC,R2                      ;SETUP POINTER TO DATA
11786 056642  012701  000002              MOV     #2,R1                   ;INIT COUNTER
11787 056646  022227  000000      2$:     CMP     (R2)+,#0                ;IS DATA CORRECT
11788 056652  001403                      BEQ     3$                      ;YES GO ON
11789 056654  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11790 056660  104003                      ERROR   +3                      ;FPP ERROR
11791                                                                     ;NO GO TO ERROR
11792 056662  077107      3$:     SOB     R1,2$                   ;ARE WE DONE
11793 056664  012701  000002              MOV     #2,R1                   ;INIT COUNTER
11794 056670  022227  177777      4$:     CMP     (R2)+,#177777           ;IS DATA CORRECT
11795 056674  001403                      BEQ     5$                      ;YES GO ON
11796 056676  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11797 056702  104003                      ERROR   +3                      ;FPP ERROR
11798                                                                     ;NO GO TO ERROR
11799 056704  077107      5$:     SOB     R1,4$                   ;ARE WE DONE
11800 056706  020327  000004              CMP     R3,#4                   ;CHECK FPS
11801 056712  001403                      BEQ     6$                      ;OK GO ON
11802 056714  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11803 056720  104003                      ERROR   +3                      ;FPP ERROR
11804                                                                     ;NO GO TO ERROR
11805 056722              6$:
11806                     ;
11809                     ;
11810                     ;---------------------------------
11811                     ;TEST  TSTF AND TSTD
11812                     ;
11813                     ;
11814 056722              TSFP26:
11815                     ;
11816 056722  005037  140054              CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11817 056726  005004                      CLR     R4                      ;SETUP TO LOAD FPS
11818 056730  170104                      LDFPS   R4                      ;SET FD=0
11819 056732  170567  000300              TSTF    TS26D0                  ; TEST INSTRUCTION
11820 056736  170203                      STFPS   R3                      ;GET FPS
11821 056740  020327  000004              CMP     R3,#4                   ;CHECK FPS
11822 056744  001403                      BEQ     1$                      ;OK GO ON
11823 056746  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11824 056752  104003                      ERROR   +3                      ;FPP ERROR
11825                                                                     ;NO GO TO ERROR
11826 056754  012704  057236      1$:     MOV     #TS26D0,R4              ;SETUP POINTERS TO DATA
11827 056760  012702  057266              MOV     #TS26D3,R2              ;
11828 056764  004767  000224              JSR     PC,CHEC26               ;CHECK IF DATA IS CORRECT
11829 056770  170537  057246              TSTF    @#TS26D1                ; TEST INSTRUCTION
```

FLOATING POINT TESTS

```
11830 056774  170203                    STFPS   R3                      ;GET FPS
11831 056776  020327  000010            CMP     R3,#10                  ;CHECK FPS
11832 057002  001403                    BEQ     2$                      ;OK GO ON
11833 057004  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11834 057010  104003                    ERROR   +3              ;FPP ERROR
11835                                                           ;NO GO TO ERROR
11836 057012  012704  057246    2$:     MOV     #TS26D1,R4              ;SETUP POINTERS TO DATA
11837 057016  012702  057276            MOV     #TS26D4,R2              ;
11838 057022  004767  000166            JSR     PC,CHEC26               ;CHECK IF DATA IS CORRECT
11839 057026  170567  000224            TSTF    TS26D2                  ; TEST INSTRUCTION
11840 057032  170203                    STFPS   R3                      ;GET FPS
11841 057034  020327  000000            CMP     R3,#0                   ;CHECK FPS
11842 057040  001403                    BEQ     3$                      ;OK GO ON
11843 057042  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11844 057046  104003                    ERROR   +3              ;FPP ERROR
11845                                                           ;NO GO TO ERROR
11846 057050  012704  057256    3$:     MOV     #TS26D2,R4              ;SETUP POINTERS TO DATA
11847 057054  012702  057306            MOV     #TS26D5,R2              ;
11848 057060  004767  000130            JSR     PC,CHEC26               ;CHECK IF DATA IS CORRECT
11849 057064  012704  000200            MOV     #200,R4                 ;SETUP TO LOAD FPS
11850 057070  170104                    LDFPS   R4                      ;SET FD=1
11851 057072  170537  057236            TSTD    @#TS26D0                ; TEST INSTRUCTION
11852 057076  170203                    STFPS   R3                      ;GET FPS
11853 057100  020327  000204            CMP     R3,#204                 ;CHECK FPS
11854 057104  001403                    BEQ     4$                      ;OK GO ON
11855 057106  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11856 057112  104003                    ERROR   +3              ;FPP ERROR
11857                                                           ;NO GO TO ERROR
11858 057114  012704  057236    4$:     MOV     #TS26D0,R4              ;SETUP POINTERS TO DATA
11859 057120  012702  057266            MOV     #TS26D3,R2              ;
11860 057124  004767  000064            JSR     PC,CHEC26               ;CHECK IF DATA IS CORRECT
11861 057130  170567  000112            TSTD    TS26D1                  ; TEST INSTRUCTION
11862 057134  170203                    STFPS   R3                      ;GET FPS
11863 057136  020327  000210            CMP     R3,#210                 ;CHECK FPS
11864 057142  001403                    BEQ     5$                      ;OK GO ON
11865 057144  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11866 057150  104003                    ERROR   +3              ;FPP ERROR
11867                                                           ;NO GO TO ERROR
11868 057152  012704  057246    5$:     MOV     #TS26D1,R4              ;SETUP POINTERS TO DATA
11869 057156  012702  057276            MOV     #TS26D4,R2              ;
11870 057162  004767  000026            JSR     PC,CHEC26               ;CHECK IF DATA IS CORRECT
11871 057166  170567  000064            TSTD    TS26D2                  ; TEST INSTRUCTION
11872 057172  170203                    STFPS   R3                      ;GET FPS
11873 057174  020327  000200            CMP     R3,#200                 ;CHECK FPS
11874 057200  001403                    BEQ     6$                      ;OK GO ON
11875 057202  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11876 057206  104003                    ERROR   +3              ;FPP ERROR
11877                                                           ;NO GO TO ERROR
11878 057210                    6$:
11879 057210  000167  000102            JMP     FIN26
11880                           ;
11881 057214  012701  000004    CHEC26: MOV     #4,R1                   ;INIT COUNTER
11882 057220  022422            1$:     CMP     (R4)+,(R2)+             ;IS DATA CORRECT
11883 057222  001403                    BEQ     2$                      ;YES GO ON
11884 057224  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11885 057230  104003                    ERROR   +3              ;FPP ERROR
11886                                                           ;NO GO TO ERROR
```

FLOATING POINT TESTS

```
11887 057232  077106                2$:     SOB     R1,1$                   ;ARE WE DONE
11888 057234  000207                        RTS     PC                      ;RETURN
11889                               ;
11890 057236  000177                TS26D0: .WORD   177
11891 057240  177777                        .WORD   177777
11892 057242  177777                        .WORD   177777
11893 057244  177777                        .WORD   177777
11894 057246  177777                TS26D1: .WORD   177777
11895 057250  000000                        .WORD   0
11896 057252  000000                        .WORD   0
11897 057254  000000                        .WORD   0
11898 057256  077777                TS26D2: .WORD   77777
11899 057260  000000                        .WORD   0
11900 057262  000000                        .WORD   0
11901 057264  000000                        .WORD   0
11902 057266  000177                TS26D3: .WORD   177
11903 057270  177777                        .WORD   177777
11904 057272  177777                        .WORD   177777
11905 057274  177777                        .WORD   177777
11906 057276  177777                TS26D4: .WORD   177777
11907 057300  000000                        .WORD   0
11908 057302  000000                        .WORD   0
11909 057304  000000                        .WORD   0
11910 057306  077777                TS26D5: .WORD   77777
11911 057310  000000                        .WORD   0
11912 057312  000000                        .WORD   0
11913 057314  000000                        .WORD   0
11914 057316  000240                FIN26:  NOP
11915                               ;
11918                               ;
11919                               ;---------------------------------
11920                               ;TEST  ABSF
11921                               ;
11922                               ;
11923 057320                        TSFP27:
11924                               ;
11925 057320  005037  140054                CLR     @#TRPFLG                ;CLEAR TRAP FLAG
11926 057324  005005                        CLR     R5                      ;SETUP TO LOAD FPS
11927 057326  170105                        LDFPS   R5                      ;SET FD=0
11928 057330  012701  000014                MOV     #12.,R1                 ;INIT COUNTER
11929 057334  012704  003162                MOV     #TSTLOC,R4                      ;SETUP POINTER TO TEST LOCATION
11930 057340  012703  057600                MOV     #TS27D0,R3              ;SETUP POINTER TO TEST VALUE
11931 057344  012324                100$:   MOV     (R3)+,(R4)+             ;INIT TEST LOCATION
11932 057346  077102                        SOB     R1,100$                 ;ARE WE DONE
11933 057350  012705  003162                MOV     #TSTLOC,R5                      ;SETUP POINTER TO DATA
11934 057354  170615                        ABSF    (R5)                    ; TEST INSTRUCTION
11935 057356  170203                        STFPS   R3                      ;GET FPS
11936 057360  020527  003162                CMP     R5,#TSTLOC                      ;IS R5 CORRECT
11937 057364  001403                        BEQ     1$                      ;YES GO ON
11938 057366  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11939 057372  104003                        ERROR   +3                  ;FPP ERROR
11940                                                                   ;NO GO TO ERROR
11941 057374  012702  057630        1$:     MOV     #TS27D3,R2              ;SETUP POINTER TO DATA
11942 057400  004767  000152                JSR     PC,CHEC27               ;CHECK IF DATA IS CORRECT
11943 057404  020327  000000                CMP     R3,#0                   ;CHECK FPS
11944 057410  001403                        BEQ     2$                      ;OK GO ON
11945 057412  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
```

FLOATING POINT TESTS

```
11946 057416  104003                    ERROR   +3              ;FPP ERROR
11947                                                           ;NO GO TO ERROR
11948 057420  012705  003172      2$:   MOV     #TSTLOC+10,R5                   ;SETUP POINTER TO DATA
11949 057424  170625                    ABSF    (R5)+                   ; TEST INSTRUCTION
11950 057426  170203                    STFPS   R3                      ;GET FPS
11951 057430  020527  003176            CMP     R5,#TSTLOC+14                   ;IS R5 CORRECT
11952 057434  001403                    BEQ     3$                      ;YES GO ON
11953 057436  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11954 057442  104003                    ERROR   +3              ;FPP ERROR
11955                                                           ;NO GO TO ERROR
11956 057444  012705  003172      3$:   MOV     #TSTLOC+10,R5                   ;SETUP POINTER TO DATA
11957 057450  012702  057640            MOV     #TS27D4,R2              ;
11958 057454  004767  000076            JSR     PC,CHEC27               ;CHECK IF DATA IS CORRECT
11959 057460  020327  000000            CMP     R3,#0                   ;CHECK FPS
11960 057464  001403                    BEQ     4$                      ;OK GO ON
11961 057466  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11962 057472  104003                    ERROR   +3              ;FPP ERROR
11963                                                           ;NO GO TO ERROR
11964 057474  012705  003162      4$:   MOV     #TSTLOC,R5                      ;SETUP POINTER TO DATA
11965 057500  170665  000020            ABSF    20(R5)                  ; TEST INSTRUCTION
11966 057504  170203                    STFPS   R3                      ;GET FPS
11967 057506  020527  003162            CMP     R5,#TSTLOC                      ;IS R5 CORRECT
11968 057512  001403                    BEQ     5$                      ;YES GO ON
11969 057514  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11970 057520  104003                    ERROR   +3              ;FPP ERROR
11971                                                           ;NO GO TO ERROR
11972 057522  012705  003202      5$:   MOV     #TSTLOC+20,R5                   ;SETUP POINTERS TO DATA
11973 057526  012702  057650            MOV     #TS27D5,R2              ;
11974 057532  004767  000020            JSR     PC,CHEC27               ;CHECK IF DATA IS CORRECT
11975 057536  020327  000004            CMP     R3,#4                   ;CHECK FPS
11976 057542  001403                    BEQ     6$                      ;OK GO ON
11977 057544  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11978 057550  104003                    ERROR   +3              ;FPP ERROR
11979                                                           ;NO GO TO ERROR
11980 057552                      6$:
11981 057552  000167  000102            JMP     FIN27
11982                                   ;
11983 057556  012701  000004      CHEC27: MOV     #4,R1                   ;INIT COUNTER
11984 057562  022522                1$:   CMP     (R5)+,(R2)+             ;IS DATA CORRECT
11985 057564  001403                    BEQ     2$                      ;YES GO ON
11986 057566  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
11987 057572  104003                    ERROR   +3              ;FPP ERROR
11988                                                           ;NO GO TO ERROR
11989 057574  077106                2$:   SOB     R1,1$                   ;ARE WE DONE
11990 057576  000207                    RTS     PC                      ;RETURN
11991                                   ;
11992 057600  177777      TS27D0: .WORD   177777
11993 057602  177777              .WORD   177777
11994 057604  177777              .WORD   177777
11995 057606  177777              .WORD   177777
11996 057610  000377      TS27D1: .WORD   377
11997 057612  175436              .WORD   175436
11998 057614  136477              .WORD   136477
11999 057616  000001              .WORD   1
12000 057620  000177      TS27D2: .WORD   177
12001 057622  175436              .WORD   175436
12002 057624  136477              .WORD   136477
```

FLOATING POINT TESTS

```
12003 057626  000001                    .WORD   1
12004 057630  077777            TS27D3: .WORD   77777
12005 057632  177777                    .WORD   177777
12006 057634  177777                    .WORD   177777
12007 057636  177777                    .WORD   177777
12008 057640  000377            TS27D4: .WORD   377
12009 057642  175436                    .WORD   175436
12010 057644  136477                    .WORD   136477
12011 057646  000001                    .WORD   1
12012 057650  000000            TS27D5: .WORD   0
12013 057652  000000                    .WORD   0
12014 057654  136477                    .WORD   136477
12015 057656  000001                    .WORD   1
12016 057660  000240            FIN27:  NOP
12017                           ;
12020                           ;
12021                           ;---------------------------------
12022                           ;TEST  ABSD
12023                           ;
12024                           ;
12025 057662                    TSFP30:
12026                           ;
12027 057662  005037  140054              CLR     @#TRPFLG                ;CLEAR TRAP FLAG
12028 057666  012705  000200              MOV     #200,R5                 ;SETUP TO LOAD FPS
12029 057672  170105                      LDFPS   R5                      ;SET FD=1
12030 057674  012701  000014              MOV     #12.,R1                 ;INIT COUNTER
12031 057700  012704  003162              MOV     #TSTLOC,R4                      ;SETUP POINTER TO TEST LOCATION
12032 057704  012703  060144              MOV     #TS30D0,R3              ;SETUP POINTER TO TEST VALUE
12033 057710  012324            100$:     MOV     (R3)+,(R4)+             ;INIT TEST LOCATION
12034 057712  077102                      SOB     R1,100$                 ;ARE WE DONE
12035 057714  012705  003162              MOV     #TSTLOC,R5                      ;SETUP POINTER TO DATA
12036 057720  170615                      ABSD    (R5)                    ; TEST INSTRUCTION
12037 057722  170203                      STFPS   R3                      ;GET FPS
12038 057724  020527  003162              CMP     R5,#TSTLOC                      ;IS R5 CORRECT
12039 057730  001403                      BEQ     1$                      ;YES GO ON
12040 057732  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12041 057736  104003                      ERROR   +3              ;FPP ERROR
12042                                     ;NO GO TO ERROR
12043 057740  012702  060174      1$:     MOV     #TS30D3,R2              ;SETUP POINTER TO DATA
12044 057744  004767  000152              JSR     PC,CHEC30               ;CHECK IF DATA IS CORRECT
12045 057750  020327  000200              CMP     R3,#200                 ;CHECK FPS
12046 057754  001403                      BEQ     2$                      ;OK GO ON
12047 057756  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12048 057762  104003                      ERROR   +3              ;FPP ERROR
12049                                     ;NO GO TO ERROR
12050 057764  012705  003172      2$:     MOV     #TSTLOC+10,R5                   ;SETUP POINTER TO DATA
12051 057770  170625                      ABSD    (R5)+                   ; TEST INSTRUCTION
12052 057772  170203                      STFPS   R3                      ;GET FPS
12053 057774  020527  003202              CMP     R5,#TSTLOC+20                   ;IS R5 CORRECT
12054 060000  001403                      BEQ     3$                      ;YES GO ON
12055 060002  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12056 060006  104003                      ERROR   +3              ;FPP ERROR
12057                                     ;NO GO TO ERROR
12058 060010  012705  003172      3$:     MOV     #TSTLOC+10,R5                   ;SETUP POINTERS TO DATA
12059 060014  012702  060204              MOV     #TS30D4,R2              ;
12060 060020  004767  000076              JSR     PC,CHEC30               ;CHECK IF DATA IS CORRECT
12061 060024  020327  000200              CMP     R3,#200                 ;CHECK FPS
```

FLOATING POINT TESTS

```
12062 060030  001403                  BEQ     4$                      ;OK GO ON
12063 060032  004737  140132          CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12064 060036  104003                  ERROR   +3              ;FPP ERROR
12065                                                         ;NO GO TO ERROR
12066 060040  012705  003162  4$:     MOV     #TSTLOC,R5                      ;SETUP POINTER TO DATA
12067 060044  170665  000020          ABSD    20(R5)                  ; TEST INSTRUCTION
12068 060050  170203                  STFPS   R3                      ;GET FPS
12069 060052  020527  003162          CMP     R5,#TSTLOC                      ;IS R5 CORRECT
12070 060056  001403                  BEQ     5$                      ;YES GO ON
12071 060060  004737  140132          CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12072 060064  104003                  ERROR   +3              ;FPP ERROR
12073                                                         ;NO GO TO ERROR
12074 060066  012705  003202  5$:     MOV     #TSTLOC+20,R5                   ;SETUP POINTERS TO DATA
12075 060072  012702  060214          MOV     #TS30D5,R2              ;
12076 060076  004767  000020          JSR     PC,CHEC30               ;CHECK IF DATA IS CORRECT
12077 060102  020327  000204          CMP     R3,#204                 ;CHECK FPS
12078 060106  001403                  BEQ     6$                      ;OK GO ON
12079 060110  004737  140132          CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12080 060114  104003                  ERROR   +3              ;FPP ERROR
12081                                                         ;NO GO TO ERROR
12082 060116                  6$:
12083 060116  000167  000102          JMP     FIN30
12084                                 ;
12085 060122  012701  000004  CHEC30: MOV     #4,R1                   ;INIT COUNTER
12086 060126  022522          1$:     CMP     (R5)+,(R2)+             ;IS DATA CORRECT
12087 060130  001403                  BEQ     2$                      ;YES GO ON
12088 060132  004737  140132          CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12089 060136  104003                  ERROR   +3              ;FPP ERROR
12090                                                         ;NO GO TO ERROR
12091 060140  077106          2$:     SOB     R1,1$                   ;ARE WE DONE
12092 060142  000207                  RTS     PC                      ;RETURN
12093                                 ;
12094 060144  177777  TS30D0: .WORD   177777
12095 060146  177777          .WORD   177777
12096 060150  177777          .WORD   177777
12097 060152  177777          .WORD   177777
12098 060154  000377  TS30D1: .WORD   377
12099 060156  175436          .WORD   175436
12100 060160  136477          .WORD   136477
12101 060162  000001          .WORD   1
12102 060164  000177  TS30D2: .WORD   177
12103 060166  175436          .WORD   175436
12104 060170  136477          .WORD   136477
12105 060172  000001          .WORD   1
12106 060174  077777  TS30D3: .WORD   77777
12107 060176  177777          .WORD   177777
12108 060200  177777          .WORD   177777
12109 060202  177777          .WORD   177777
12110 060204  000377  TS30D4: .WORD   377
12111 060206  175436          .WORD   175436
12112 060210  136477          .WORD   136477
12113 060212  000001          .WORD   1
12114 060214  000000  TS30D5: .WORD   0
12115 060216  000000          .WORD   0
12116 060220  000000          .WORD   0
12117 060222  000000          .WORD   0
12118 060224  000240  FIN30:  NOP
```

FLOATING POINT TESTS

```
12119                                   ;
12122                                   ;
12123                                   ;---------------------------------
12124                                   ;TEST  FDST SOP MODE 2 GR7
12125                                   ;
12126 060226                            TSFP31:
12127                                   ;
12128 060226  005037  140054                    CLR     @#TRPFLG                ;CLEAR TRAP FLAG
12129 060232  013746  000004                    MOV     @#4,-(SP)               ;SAVE TIMEOUT VECTOR
12130 060236  012737  060354  000004            MOV     #TSF31,@#4              ;SETUP NEW VECTOR
12131 060244  012702  000200                    MOV     #200,R2                 ;SETUP TO LOAD FPS
12132 060250  170102                            LDFPS   R2                      ;SET FD=1
12133 060252  170527  000005            TSD31:  TSTD    #5                      ; TEST INSTRUCTION
12134 060256  000240                            NOP
12135 060260  000240                            NOP
12136 060262  000240                            NOP
12137 060264  170203                            STFPS   R3                      ;GET FPS
12138 060266  020327  000204                    CMP     R3,#204                 ;CHECK FPS
12139 060272  001403                            BEQ     1$                      ;OK GO ON
12140 060274  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12141 060300  104003                            ERROR   .+3                     ;FPP ERROR
12142                                                                           ;NO GO TO ERROR
12143 060302  012702  060254            1$:     MOV     #TSD31+2,R2             ;SETUP POINTER TO DATA
12144 060306  022227  000005                    CMP     (R2)+,#5                ;IS DATA CORRECT
12145 060312  001403                            BEQ     2$                      ;YES GO ON
12146 060314  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12147 060320  104003                            ERROR   .+3                     ;FPP ERROR
12148                                                                           ;NO GO TO ERROR
12149 060322  012701  000003            2$:     MOV     #3,R1                   ;INIT COUNTER
12150 060326  022227  000240            3$:     CMP     (R2)+,#240              ;IS DATA CORRECT
12151 060332  001403                            BEQ     4$                      ;YES GO ON
12152 060334  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12153 060340  104003                            ERROR   .+3                     ;FPP ERROR
12154                                                                           ;NO GO TO ERROR
12155 060342  077107                    4$:     SOB     R1,3$                   ;ARE WE DONE
12156 060344  012637  000004                    MOV     (SP)+,@#4               ;RESTORE VECTOR
12157 060350  000167  000010                    JMP     FIN31
12158                                   ;
12159 060354  004737  140132            TSF31:  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12160 060360  104003                            ERROR   .+3                     ;FPP ERROR
12161                                                                           ;ODD ADDRESS TRAP
12162 060362  000006                            RTT                             ;RETURN
12163                                   ;
12164 060364  000240                    FIN31:  NOP
12165                                   ;
12168                                   ;
12169                                   ;---------------------------------
12170                                   ;TEST  NEGF
12171                                   ;
12172                                   ;
12173 060366                            TSFP32:
12174                                   ;
12175 060366  005037  140054                    CLR     @#TRPFLG                ;CLEAR TRAP FLAG
12176 060372  005005                            CLR     R5                      ;SETUP TO LOAD FPS
12177 060374  170105                            LDFPS   R5                      ;SET FD=0
12178 060376  012701  000014                    MOV     #12.,R1                 ;INIT COUNTER
12179 060402  012704  003162                    MOV     #TSTLOC,R4                      ;SETUP POINTER TO TEST LOCATION
```

FLOATING POINT TESTS

```
12180 060406  012703  060576            MOV     #TS32D0,R3              ;SETUP POINTER TO TEST VALUE
12181 060412  012324              100$:   MOV     (R3)+,(R4)+             ;INIT TEST LOCATION
12182 060414  077102                      SOB     R1,100$                 ;ARE WE DONE
12183 060416  170767  122540              NEGF    TSTLOC                  ; TEST INSTRUCTION
12184 060422  170203                      STFPS   R3                      ;GET FPS
12185 060424  012705  003162              MOV     #TSTLOC,R5                      ;SETUP POINTERS TO DATA
12186 060430  012702  060626              MOV     #TS32D3,R2              ;
12187 060434  004767  000114              JSR     PC,CHEC32               ;CHECK IF DATA IS CORRECT
12188 060440  020327  000000              CMP     R3,#0                   ;CHECK FPS
12189 060444  001403                      BEQ     1$                      ;YES GO ON
12190 060446  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12191 060452  104003                      ERROR   +3              ;FPP ERROR
12192                                                             ;NO GO TO ERROR
12193 060454  170767  122512        1$:   NEGF    TSTLOC+10                       ; TEST INSTRUCTION
12194 060460  170203                      STFPS   R3                      ;GET FPS
12195 060462  012705  003172              MOV     #TSTLOC+10,R5                   ;SETUP POINTERS TO DATA
12196 060466  012702  060636              MOV     #TS32D4,R2              ;
12197 060472  004767  000056              JSR     PC,CHEC32               ;CHECK IF DATA IS CORRECT
12198 060476  020327  000010              CMP     R3,#10                  ;CHECK FPS
12199 060502  001403                      BEQ     2$                      ;OK GO ON
12200 060504  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12201 060510  104003                      ERROR   +3              ;FPP ERROR
12202                                                             ;NO GO TO ERROR
12203 060512  170767  122464        2$:   NEGF    TSTLOC+20                       ; TEST INSTRUCTION
12204 060516  170203                      STFPS   R3                      ;GET FPS
12205 060520  012705  003202              MOV     #TSTLOC+20,R5                   ;SETUP POINTERS TO DATA
12206 060524  012702  060646              MOV     #TS32D5,R2              ;
12207 060530  004767  000020              JSR     PC,CHEC32               ;CHECK IF DATA IS CORRECT
12208 060534  020327  000004              CMP     R3,#4                   ;CHECK FPS
12209 060540  001403                      BEQ     3$                      ;OK GO ON
12210 060542  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12211 060546  104003                      ERROR   +3              ;FPP ERROR
12212                                                             ;NO GO TO ERROR
12213 060550                        3$:
12214 060550  000167  000102              JMP     FIN32
12215                             ;
12216 060554  012701  000004      CHEC32: MOV     #4,R1                   ;INIT COUNTER
12217 060560  022522              1$:     CMP     (R5)+,(R2)+             ;IS DATA CORRECT
12218 060562  001403                      BEQ     2$                      ;YES GO ON
12219 060564  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12220 060570  104003                      ERROR   +3              ;FPP ERROR
12221                                                             ;NO GO TO ERROR
12222 060572  077106              2$:     SOB     R1,1$                   ;ARE WE DONE
12223 060574  000207                      RTS     PC                      ;RETURN
12224                             ;
12225 060576  170000      TS32D0: .WORD   170000
12226 060600  003541              .WORD   3541
12227 060602  177777              .WORD   177777
12228 060604  172710              .WORD   172710
12229 060606  070000      TS32D1: .WORD   70000
12230 060610  003541              .WORD   3541
12231 060612  177777              .WORD   177777
12232 060614  172710              .WORD   172710
12233 060616  000177              .WORD   177
12234 060620  100000              .WORD   100000
12235 060622  177777              .WORD   177777
12236 060624  177007              .WORD   177007
```

FLOATING POINT TESTS

```
12237 060626  070000                  TS32D3: .WORD   70000
12238 060630  003541                          .WORD   3541
12239 060632  177777                          .WORD   177777
12240 060634  172710                          .WORD   172710
12241 060636  170000                  TS32D4: .WORD   170000
12242 060640  003541                          .WORD   3541
12243 060642  177777                          .WORD   177777
12244 060644  172710                          .WORD   172710
12245 060646  000000                  TS32D5: .WORD   0
12246 060650  000000                          .WORD   0
12247 060652  177777                          .WORD   177777
12248 060654  177007                          .WORD   177007
12249 060656  000240                  FIN32:  NOP
12250                                 ;
12253                                 ;
12254                                 ;---------------------------------
12255                                 ;TEST  NEGD
12256                                 ;
12257 060660                          TSFP33:
12258                                 ;
12259 060660  005037  140054                  CLR     @#TRPFLG                ;CLEAR TRAP FLAG
12260 060664  012705  000200                  MOV     #200,R5                 ;SETUP TO LOAD FPS
12261 060670  170105                          LDFPS   R5                      ;SET FD=1
12262 060672  012701  000014                  MOV     #12.,R1                 ;INIT COUNTER
12263 060676  012704  003162                  MOV     #TSTLOC,R4                      ;SETUP POINTER TO TEST LOCATION
12264 060702  012703  061072                  MOV     #TS33D0,R3              ;SETUP POINTER TO TEST VALUE
12265 060706  012324                  100$:   MOV     (R3)+,(R4)+             ;INIT TEST LOCATION
12266 060710  077102                          SOB     R1,100$                 ;ARE WE DONE
12267 060712  170767  122244                  NEGD    TSTLOC                  ; TEST INSTRUCTION
12268 060716  170203                          STFPS   R3                      ;GET FPS
12269 060720  012705  003162                  MOV     #TSTLOC,R5                      ;SETUP POINTERS TO DATA
12270 060724  012702  061122                  MOV     #TS33D3,R2              ;
12271 060730  004767  000114                  JSR     PC,CHEC33               ;CHECK IF DATA IS CORRECT
12272 060734  020327  000200                  CMP     R3,#200                 ;CHECK FPS
12273 060740  001403                          BEQ     1$                      ;OK GO ON
12274 060742  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12275 060746  104003                          ERROR   +3              ;FPP ERROR
12276                                                                 ;NO GO TO ERROR
12277 060750  170767  122216          1$:     NEGD    TSTLOC+10                       ; TEST INSTRUCTION
12278 060754  170203                          STFPS   R3                      ;GET FPS
12279 060756  012705  003172                  MOV     #TSTLOC+10,R5                   ;SETUP POINTERS TO DATA
12280 060762  012702  061132                  MOV     #TS33D4,R2              ;
12281 060766  004767  000056                  JSR     PC,CHEC33               ;CHECK IF DATA IS CORRECT
12282 060772  020327  000210                  CMP     R3,#210                 ;CHECK FPS
12283 060776  001403                          BEQ     2$                      ;OK GO ON
12284 061000  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12285 061004  104003                          ERROR   +3              ;FPP ERROR
12286                                                                 ;NO GO TO ERROR
12287 061006  170767  122170          2$:     NEGD    TSTLOC+20                       ; TEST INSTRUCTION
12288 061012  170203                          STFPS   R3                      ;GET FPS
12289 061014  012705  003202                  MOV     #TSTLOC+20,R5                   ;SETUP POINTERS TO DATA
12290 061020  012702  061142                  MOV     #TS33D5,R2              ;
12291 061024  004767  000020                  JSR     PC,CHEC33               ;CHECK IF DATA IS CORRECT
12292 061030  020327  000204                  CMP     R3,#204                 ;CHECK FPS
12293 061034  001403                          BEQ     3$                      ;OK GO ON
12294 061036  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12295 061042  104003                          ERROR   +3              ;FPP ERROR
```

FLOATING POINT TESTS

```
12296                                                                   ;NO GO TO ERROR
12297 061044                            3$:
12298 061044  000167  000102                      JMP     FIN33
12299                                   ;
12300 061050  012701  000004            CHEC33:   MOV     #4,R1                 ;INIT COUNTER
12301 061054  022522                    1$:       CMP     (R5)+,(R2)+           ;IS DATA CORRECT
12302 061056  001403                              BEQ     2$                    ;YES GO ON
12303 061060  004737  140132                      CALL    @#DETFPA              ;DETERMINE FLOATING POINT FAULT. $$$
12304 061064  104003                              ERROR   +3                    ;FPP ERROR
12305                                                                   ;NO GO TO ERROR
12306 061066  077106                    2$:       SOB     R1,1$                 ;ARE WE DONE
12307 061070  000207                              RTS     PC                    ;RETURN
12308                                   ;
12309 061072  170000                    TS33D0:   .WORD   170000
12310 061074  003541                              .WORD   3541
12311 061076  177777                              .WORD   177777
12312 061100  172710                              .WORD   172710
12313 061102  070000                    TS33D1:   .WORD   70000
12314 061104  003541                              .WORD   3541
12315 061106  177777                              .WORD   177777
12316 061110  172710                              .WORD   172710
12317 061112  000177                    TS33D2:   .WORD   177
12318 061114  100000                              .WORD   100000
12319 061116  177777                              .WORD   177777
12320 061120  177007                              .WORD   177007
12321 061122  070000                    TS33D3:   .WORD   70000
12322 061124  003541                              .WORD   3541
12323 061126  177777                              .WORD   177777
12324 061130  172710                              .WORD   172710
12325 061132  170000                    TS33D4:   .WORD   170000
12326 061134  003541                              .WORD   3541
12327 061136  177777                              .WORD   177777
12328 061140  172710                              .WORD   172710
12329 061142  000000                    TS33D5:   .WORD   0
12330 061144  000000                              .WORD   0
12331 061146  000000                              .WORD   0
12332 061150  000000                              .WORD   0
12333 061152  000240                    FIN33:    NOP
12334                                   ;
12337                                   ;
12338                                   ;---------------------------------
12339                                   ;TEST LDD MODE 0, ILLEGAL AC7
12340                                   ;
12341                                             ;
12342 061154                            MFSRCM0:
12343
12344                                   ;
12345 061154  012704  047600                      MOV     #47600,R4             ;SETUP FPP STATUS
12346 061160  170104                              LDFPS   R4                    ;LOAD FP STATUS
12347 061162  012702  003122                      MOV     #RECFEC,R2            ;POINT TO RECEIVED FEC MEMORY
12348 061166  172407                    1$:       LDD     R7,AC0                ;*TEST INSTRUCTION
12349                                                                           ;LOAD AC0 FROM ILLEGAL AC7
12350 061170  170201                              STFPS   R1                    ;SAVE FPP STATUS
12351 061172  022701  147600                      CMP     #147600,R1            ;VERIFY FER BIT SET
12352 061176  001403                              BEQ     2$                    ;BRANCH IF GOOD ERROR CONDITION
12353 061200  004737  140132                      CALL    @#DETFPA              ;DETERMINE FLOATING POINT FAULT. $$$
12354 061204  104003                              ERROR   +3                    ;FPP ERROR
```

FLOATING POINT TESTS

```
12355
12356 061206  170312                2$:     STST    (R2)                    ;THE FER BIT DIDNT SET
12357 061210  022722  000002                CMP     #2,(R2)+                        ;SAVE FEC AND FEA
12358 061214  001403                        BEQ     3$                              ;VERIFY FEC CONTENTS
12359 061216  004737  140132                CALL    @#DETFPA                        ;BRANCH IF GOOD
12360 061222  104003                        ERROR   +3                              ;DETERMINE FLOATING POINT FAULT. $$$
12361                                                                       ;FPP ERROR
12362 061224  022722  061166          3$:   CMP     #1$,(R2)+               ;FEC NE 2 (OPCODE ERROR)
12363 061230  001403                        BEQ     4$                              ;VERFIY FEA CONTENTS
12364 061232  004737  140132                CALL    @#DETFPA                        ;BRANCH FI GOOD
12365 061236  104003                        ERROR   +3                              ;DETERMINE FLOATING POINT FAULT. $$$
12366                                                                       ;FPP ERROR
12367 061240                          4$:                                   ;FEA NOT CORRECT ERROR ADDRESS
12370                                 ;
12371                                 ;---------------------------------
12372                                 ;TEST LDD MODE2
12373                                 ;
12374 061240                          MLDDM2:
12375                                 ;
12376 061240  012701  003142                MOV     #RECDST,R1                      ;POINT TO RECEIVED DATA LOCATION
12377 061244  012704  003234                MOV     #TAB1,R4                        ;POINT TO GOOD DATA
12378 061250  012702  047750                MOV     #47750,R2                       ;LOAD GOOD STATUS
12379 061254  170102                        LDFPS   R2                              ;LOAD FPP STATUS - DOUBLE,ID
12380 061256  172424                        LDD     (R4)+,AC0                       ;*TEST INSTRUCTION - MODE 2
12381 061260  170203                        STFPS   R3                              ;SAVE TEST FPP STATUS
12382 061262  174021                        STD     AC0,(R1)+                       ;SAVE TEST RESULT MODE 2
12383 061264  020203                        CMP     R2,R3                           ;VERIFY FPP STATUS
12384 061266  001403                        BEQ     1$                              ;BRANCH IF GOOD
12385 061270  004737  140132                CALL    @#DETFPA                        ;DETERMINE FLOATING POINT FAULT. $$$
12386 061274  104003                        ERROR   +3                      ;FPP ERROR
12387                                                                       ;BAD FPP STSTUS
12388 061276  022704  003244          1$:   CMP     #TAB1+10,R4                             ;VERFIY AUTO-INCR
12389 061302  001403                        BEQ     2$                              ;BRANCH IF GOOD
12390 061304  004737  140132                CALL    @#DETFPA                        ;DETERMINE FLOATING POINT FAULT. $$$
12391 061310  104003                        ERROR   +3                      ;FPP ERROR
12392                                                                       ;BAD AUTO-INCR
12393 061312  012704  003234          2$:   MOV     #TAB1,R4                                ;POINT TO RECEIVED DATA
12394 061316  162701  000010                SUB     #10,R1                          ;RETURN R1 TO PROPER VALUE
12395 061322  004767  056552                JSR     P7,DATVER                       ;VERFIY DATA FROM FPP
12396 061326  005767  121566                TST     COUNT                           ;SEE IF COUNTER=0
12397 061332  001403                        BEQ     3$                              ;BRANCH IF GOOD COMPARE
12398 061334  004737  140132                CALL    @#DETFPA                        ;DETERMINE FLOATING POINT FAULT. $$$
12399 061340  104003                        ERROR   +3              ;FPP ERROR                                      ;BAD DATA FROM FPP
12400 061342                          3$:
12401                                 ;
12404                                 ;
12405                                 ;---------------------------------
12406                                 ;TEST LDD MODE 3
12407                                 ;
12408 061342                          MLDDM3:
12409                                 ;
12410 061342  012737  003142  003164        MOV     #RECDST,@#TSTLOC+2                      ;POINT TO RECEIVED DATA LOCATION
12411 061350  012701  003164                MOV     #TSTLOC+2,R1                    ;SETUP STD IN MODE 3
12412 061354  012737  003244  003162        MOV     #TAB2,@#TSTLOC          ;POINT TO DATA TABLE
12413 061362  012704  003162                MOV     #TSTLOC,R4                      ;POINT TO GOOD DATA
12414 061366  012702  047750                MOV     #47750,R2               ;LOAD GOOD STATUS
12415 061372  170102                        LDFPS   R2                      ;LOAD FPP STATUS - DOUBLE,ID
```

FLOATING POINT TESTS

```
12416 061374  172434                    LDD     @(R4)+,AC0              ;*TEST INSTRUCTION - MODE 2
12417 061376  170203                    STFPS   R3                      ;SAVE TEST FPP STATUS
12418 061400  174031                    STD     AC0,@(R1)+              ;SAVE TEST RESULT IN MODE 3
12419 061402  022703  047740            CMP     #47740,R3               ;VERIFY FPP STATUS
12420 061406  001403                    BEQ     1$                      ;BRANCH IF GOOD
12421 061410  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12422 061414  104003                    ERROR   +3                      ;FPP ERROR
12423                                                                   ;BAD FPP STSTUS
12424 061416  022704  003164      1$:   CMP     #TSTLOC+2,R4            ;VERFIY AUTO-INCR
12425 061422  001403                    BEQ     2$                      ;BAD AUTO-DEC ON LDD
12426 061424  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12427 061430  104003                    ERROR   +3                      ;FPP ERROR
12428                                                                   ;BAD AUTO-INC
12429 061432  022701  003166      2$:   CMP     #TSTLOC+4,R1            ; TEST STD AUTO-INC
12430 061436  001403                    BEQ     3$                      ;BRANCH IF GOOD
12431 061440  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12432 061444  104003                    ERROR   +3                      ;FPP ERROR
12433                                                                   ;BAD AUTO-INCR
12434 061446  012704  003244      3$:   MOV     #TAB2,R4                ;POINT TO RECEIVED DATA
12435 061452  012701  003142            MOV     #RECDST,R1              ;POINT TO RECEIVED DATA
12436 061456  004767  056416            JSR     R7,DATVER               ;VERFIY DATA FROM FPP
12437 061462  005767  121432            TST     COUNT                   ;SEE IF COUNTER=0
12438 061466  001403                    BEQ     4$                      ;BRANCH IF GOOD COMPARE
12439 061470  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12440 061474  104003                    ERROR   +3                      ;FPP ERROR
12441                                                                   ;BAD DATA FROM FPP
12442 061476                      4$:
12443                                   ;
12446                                   ;
12447                                   ;---------------------------------
12448                                   ;TEST LDF, STD MODE 4
12449                                   ;
12450 061476                      MLDDM4:
12451                                   ;
12452 061476  012701  003146            MOV     #RECDST+4,R1            ;POINT TO RECEIVED DATA LOCATION
12453 061502  012704  003260            MOV     #TAB3+4,R4              ;POINT TO GOOD DATA
12454 061506  012705  003314            MOV     #TAB6,R5                ;CLEAR OUT AC0
12455 061512  170127  000200            LDFPS   #200                    ;SET TO DOUBLE
12456 061516  172415                    LDD     (R5),AC0                ;AC0=0
12457 061520  012702  047550            MOV     #47550,R2               ;LOAD GOOD STATUS FLOATING
12458 061524  170102                    LDFPS   R2                      ;LOAD FPP STATUS - DOUBLE,ID
12459 061526  172444                    LDF     -(R4),AC0               ;*TEST INSTRUCTION - MODE 4
12460 061530  170203                    STFPS   R3                      ;SAVE TEST FPP STATUS
12461 061532  012702  047750            MOV     #47750,R2               ;SET TO DOUBLE MODE
12462 061536  170102                    LDFPS   R2                      ;SET FPP TO DOUBLE
12463 061540  174041                    STD     AC0,-(R1)               ;SAVE TEST RESULT
12464 061542  022703  047540            CMP     #47540,R3               ;VERIFY FPP STATUS
12465 061546  001403                    BEQ     1$                      ;BRANCH IF GOOD
12466 061550  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12467 061554  104003                    ERROR   +3                      ;FPP ERROR
12468                                                                   ;BAD FPP STSTUS
12469 061556  022704  003254      1$:   CMP     #TAB3,R4                ;VERFIY AUTO-DEC
12470 061562  001403                    BEQ     2$                      ;BRANCH IF GOOD
12471 061564  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12472 061570  104003                    ERROR   +3                      ;FPP ERROR
12473                                                                   ;BAD AUTO-INCR
12474 061572  012704  003254      2$:   MOV     #TAB3,R4                ;POINT TO RECEIVED DATA
```

FLOATING POINT TESTS

```
12475 061576  004767  056276                JSR     R7,DATVER               ;VERFIY DATA FROM FPP
12476 061602  005767  121312                TST     COUNT                   ;SEE IF COUNTER=0
12477 061606  001403                        BEQ     3$                      ;BRANCH IF GOOD COMPARE
12478 061610  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12479 061614  104003                        ERROR   .3                  ;FPP ERROR
12480                                                                     ;BAD DATA FROM FPP
12481 061616                        3$:
12482                               ;
12485                               ;
12486                               ;---------------------------------
12487                               ;TEST LDD MODE 5
12488                               ;
12489 061616                        MLDDM5:
12490                               ;
12491 061616  012701  003142                MOV     #RECDST,R1              ;POINT TO RECEIVED DATA LOCATION
12492 061622  012704  003164                MOV     #TSTLOC+2,R4                    ;POINT TO GOOD DATA
12493 061626  012737  003234  003162        MOV     #TAB1,@#TSTLOC                  ;SET UP MODE 5 POINTER TO DATA
12494 061634  012702  047750                MOV     #47750,R2               ;LOAD GOOD STATUS
12495 061640  170102                        LDFPS   R2                      ;LOAD FPP STATUS - DOUBLE,ID
12496 061642  172454                        LDD     @-(R4),AC0              ;*TEST INSTRUCTION - MODE 5
12497 061644  170203                        STFPS   R3                      ;SAVE TEST FPP STATUS
12498 061646  174011                        STD     AC0,(R1)                ;SAVE TEST RESULT
12499 061650  020203                        CMP     R2,R3                   ;VERIFY FPP STATUS
12500 061652  001403                        BEQ     1$                      ;BRANCH IF GOOD
12501 061654  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12502 061660  104003                        ERROR   .3                  ;FPP ERROR
12503                                                                     ;BAD FPP STATUS
12504 061662  022704  003162        1$:     CMP     #TSTLOC,R4                      ;VERFIY AUTO-DEC
12505 061666  001403                        BEQ     2$                      ;BRANCH IF GOOD
12506 061670  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12507 061674  104003                        ERROR   .3                  ;FPP ERROR
12508                                                                     ;BAD AUTO-DEC
12509 061676  012704  003234        2$:     MOV     #TAB1,R4                        ;POINT TO EXPECTED DATA
12510 061702  004767  056172                JSR     R7,DATVER               ;VERFIY DATA FROM FPP
12511 061706  005767  121206                TST     COUNT                   ;SEE IF COUNTER=0
12512 061712  001403                        BEQ     3$                      ;BRANCH IF GOOD COMPARE
12513 061714  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12514 061720  104003                        ERROR   .3                  ;FPP ERROR
12515                                                                     ;BAD DATA FROM FPP
12516 061722                        3$:
12519                               ;
12520                               ;---------------------------------
12521                               ;TEST LDD MODE 6
12522                               ;
12523 061722                        MLDDM6:
12524                               ;
12525 061722  012701  003342                MOV     #RECDST+200,R1          ;POINT TO RECEIVED DATA LOCATION
12526 061726  012704  003044                MOV     #TAB2-200,R4                    ;SETUP R4 FOR MODE 6
12527 061732  012702  047750                MOV     #47750,R2               ;LOAD GOOD STATUS
12528 061736  170102                        LDFPS   R2                      ;LOAD FPP STATUS - DOUBLE,ID
12529 061740  172464  000200                LDD     200(R4),AC0             ;LDD MODE 6
12530 061744  170203                        STFPS   R3                      ;SAVE TEST FPP STATUS
12531 061746  174061  177600                STD     AC0,-200(R1)                    ;SAVE TEST RESULT
12532 061752  022703  047740                CMP     #47740,R3                       ;VERIFY FPP STATUS
12533 061756  001403                        BEQ     1$                      ;BRANCH IF GOOD
12534 061760  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12535 061764  104003                        ERROR   .3                  ;FPP ERROR
```

FLOATING POINT TESTS

```
12536
12537 061766  162701  000200          1$:     SUB     #200,R1                 ;BAD FPP STATUS
12538 061772  062704  000200          2$:     ADD     #200,R4                         ;R1=RECDST
12539 061776  004767  056076                  JSR     R7,DATVER                       ;POINT TO EXPECTED DATA
12540 062002  005767  121112                  TST     COUNT                           ;VERFIY DATA FROM FPP
12541 062006  001403                          BEQ     3$                              ;SEE IF COUNTER=0
12542 062010  004737  140132                  CALL    @#DETFPA                        ;BRANCH IF GOOD COMPARE
12543 062014  104003                          ERROR   .3                              ;DETERMINE FLOATING POINT FAULT. $$$
12544                                                                         ;FPP ERROR
12545 062016                          3$:                                   ;BAD DATA FROM FPP
12548                                 ;
12549                                 ;---------------------------------
12550                                 ;TEST LDD MODE 7
12551                                 ;
12552 062016                          MLDDM7:
12553                                 ;
12554 062016  012701  003142                  MOV     #RECDST,R1                      ;POINT TO RECEIVED DATA LOCATION
12555 062022  005004                          CLR     R4                              ;R4=0
12556 062024  012727  003234  003162          MOV     #TAB1,#TSTLOC                   ;POINTER FOR MODE 7 GOOD DATA
12557 062032  012702  047750                  MOV     #47750,R2                       ;LOAD GOOD STATUS
12558 062036  170102                          LDFPS   R2                              ;LOAD FPP STATUS - DOUBLE,ID
12559 062040  172474  003162                  LDD     @TSTLOC(R4),AC0                 ;*TEST INSTRUCTION - MODE 7
12560 062044  170203                          STFPS   R3                              ;SAVE TEST FPP STATUS
12561 062046  174011                          STD     AC0,(R1)                                ;SAVE TEST RESULT
12562 062050  020203                          CMP     R2,R3                           ;VERIFY FPP STATUS
12563 062052  001403                          BEQ     1$                              ;BRANCH IF GOOD
12564 062054  004737  140132                  CALL    @#DETFPA                        ;DETERMINE FLOATING POINT FAULT. $$$
12565 062060  104003                          ERROR   .3                      ;FPP ERROR
12566                                                                         ;BAD FPP STATUS
12567 062062  005704                  1$:     TST     R4                              ;VERFIY CONTENTS OF R4
12568 062064  001403                          BEQ     2$                              ;BRANCH IF GOOD
12569 062066  004737  140132                  CALL    @#DETFPA                        ;DETERMINE FLOATING POINT FAULT. $$$
12570 062072  104003                          ERROR   .3                      ;FPP ERROR
12571                                                                         ;BAD R4
12572 062074  012704  003234          2$:     MOV     #TAB1,R4                                ;POINT TO RECEIVED DATA
12573 062100  004767  055774                  JSR     R7,DATVER                       ;VERFIY DATA FROM FPP
12574 062104  005767  121010                  TST     COUNT                           ;SEE IF COUNTER=0
12575 062110  001403                          BEQ     3$                              ;BRANCH IF GOOD COMPARE
12576 062112  004737  140132                  CALL    @#DETFPA                        ;DETERMINE FLOATING POINT FAULT. $$$
12577 062116  104003                          ERROR   .3                      ;FPP ERROR
12578                                                                         ;BAD DATA FROM FPP
12579 062120                          3$:
12582                                 ;
12583                                 ;---------------------------------
12584                                 ;TEST LDD MODE 27 - ONLY 16 BITS ARE LOADED OR STORED
12585                                 ;
12586 062120                          MLDM27:
12587                                 ;
12588 062120  012701  003142                  MOV     #RECDST,R1                      ;POINT TO RECEIVED DATA LOCATION
12589 062124  012704  003274                  MOV     #TAB5,R4                                ;POINT TO GOOD DATA
12590 062130  012702  047750                  MOV     #47750,R2                       ;LOAD GOOD STATUS
12591 062134  005005                          CLR     R5                              ;R5=0
12592 062136  170102                          LDFPS   R2                              ;LOAD FPP STATUS - DOUBLE,ID
12593 062140  172427  043243                  LDD     #5205,AC0                       ;*TEST INSTRUCTION - MODE 27
12594 062144  005205                          INC     R5
12595 062146  005205                          INC     R5
12596 062150  005205                          INC     R5                              ; TEST PROPER PC PATH
```

FLOATING POINT TESTS

```
12597 062152  022705  000003            CMP     #3,R5                   ;VERIFY ONLY 3 PC INCREMENT
12598 062156  001403                    BEQ     1$                      ;BRANCH IF PROPER PC ACTION
12599 062160  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12600 062164  104003                    ERROR   +3              ;FPP ERROR
12601                                                           ;BAD MODE 27 LOAD
12602 062166  170203          1$:       STFPS   R3                      ;SAVE TEST FPP STATUS
12603 062170  174011                    STD     AC0,(R1)                        ;SAVE TEST RESULT
12604 062172  022703  047740            CMP     #47740,R3                       ;VERIFY FPP STATUS
12605 062176  001403                    BEQ     2$                      ;BRANCH IF GOOD
12606 062200  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12607 062204  104003                    ERROR   +3              ;FPP ERROR
12608                                                           ;BAD FPP STATUS
12609 062206  004767  055666    2$:     JSR     R7,DATVER               ;VERFIY DATA FROM FPP
12610 062212  005767  120702            TST     COUNT                   ;SEE IF COUNTER=0
12611 062216  001403                    BEQ     3$                      ;BRANCH IF GOOD COMPARE
12612 062220  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12613 062224  104003                    ERROR   +3              ;FPP ERROR
12614                                                           ;BAD DATA FROM FPP
12615 062226                    3$:
12618                           ;
12619                           ;---------------------------------
12620                           ;TEST ADDF, ADDD, SUBF, SUBD - AC0=0 FSRC=0;
12621                           ;
12622 062226                    MNNRM1:
12623                           ;
12624 062226  012704  003314            MOV     #TAB6,R4                ;POINT TO FSRC TEST DATA
12625 062232  005067  120710            CLR     RECDST+4                ;CLEAR OUT RECEIVED DATA TABLE
12626 062236  005067  120706            CLR     RECDST+6                ;
12627 062242  012702  040000            MOV     #40000,R2               ;SET UP GOOD STATUS
12628 062246  170102                    LDFPS   R2                      ;LOAD FPP STATUS, FLOATING
12629 062250  172414                    LDF     (R4),AC0                ;LOAD AC0 WITH 0
12630 062252  172014                    ADDF    (R4),AC0                ;0+0
12631 062254  170203                    STFPS   R3                      ;SAVE STATUS
12632 062256  022703  040004            CMP     #40004,R3               ;VERIFY STATUS
12633 062262  001403                    BEQ     1$                      ;BRANCH IF GOOD
12634 062264  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12635 062270  104003                    ERROR   +3              ;FPP ERROR
12636                                                           ;BAD FPP STATUS
12637 062272  012701  003142    1$:     MOV     #RECDST,R1              ;POINT TO RECEIVERD DATA
12638 062276  174011                    STF     AC0,(R1)                ;SAVE DATA
12639 062300  004767  055574            JSR     R7,DATVER               ;VERIFY DATA
12640 062304  005767  120610            TST     COUNT                   ;
12641 062310  001403                    BEQ     2$                      ;BRANCH IF GOOD
12642 062312  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12643 062316  104003                    ERROR   +3              ;FPP ERROR
12644                                                           ;BAD DATA IN AC0
12645 062320  012702  040200    2$:     MOV     #40200,R2               ;LOAD FLOATING STATUS
12646 062324  170102                    LDFPS   R2                      ;
12647 062326  172414                    LDD     (R4),AC0                ;LOAD AC0 WITH 0
12648 062330  172014                    ADDD    (R4),AC0                ;*TEST INSTRUCTION
12649 062332  174011                    STD     AC0,(R1)                ;SAVE DATA
12650 062334  170203                    STFPS   R3                      ;SAVE FPS
12651 062336  022703  040204            CMP     #40204,R3               ;VERFIY STATUS
12652 062342  001403                    BEQ     3$                      ;BRANCH IF GOOD
12653 062344  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12654 062350  104003                    ERROR   +3              ;FPP ERROR
12655                                                           ;BAD FPS
```

FLOATING POINT TESTS

```
12656 062352  004767  055522          3$:     JSR     R7,DATVER                       ;VERFIY DATA
12657 062356  005737  003120                  TST     @#COUNT                 ;VERIFY RESULT
12658 062362  001403                          BEQ     44$                     ;BRANCH IF GOOD
12659 062364  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12660 062370  104003                          ERROR   .3                      ;FPP ERROR
12661                                                                         ;BAD ACO
12662 062372  172414          44$:    LDD     (R4),AC0                        ;SETUP DATA
12663 062374  173014                  SUBD    (R4),AC0                        ;*TEST INSTRUCTION
12664 062376  170203                  STFPS   R3                              ;SAVE STATUS
12665 062400  022703  040204                  CMP     #40204,R3                       ;VERFIY STATUS
12666 062404  001403                          BEQ     4$                      ;BRANCH IF GOOD
12667 062406  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12668 062412  104003                          ERROR   .3                      ;FPP ERROR
12669                                                                         ;BAD FPS
12670 062414  174011          4$:     STD     AC0,(R1)                        ;SAVE AC0 DATA
12671 062416  004767  055456                  JSR     R7,DATVER                       ;VERFIY DATA
12672 062422  005767  120472                  TST     COUNT                   ;
12673 062426  001403                          BEQ     5$                      ;BRANCH IF GOOD
12674 062430  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12675 062434  104003                          ERROR   .3                      ;FPP ERROR
12676                                                                         ;BAD AC0
12677 062436  170127  000000          5$:     LDFPS   #0                      ;STORE FPP STATUS
12678 062442  172414                  LDD     (R4),AC0                        ;LOAD AC0
12679 062444  173014                  SUBF    (R4),AC0                        ;0-0
12680 062446  170203                  STFPS   R3                              ;SAVE STATUS
12681 062450  174011                  STD     AC0,(R1)                        ;SAVE AC0
12682 062452  022703  000004                  CMP     #4,R3                   ;VERFIY STATUS
12683 062456  001403                          BEQ     6$                      ;BRANCH IF GOOD
12684 062460  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12685 062464  104003                          ERROR   .3                      ;FPP ERROR
12686                                                                         ;BAD FPS
12687 062466  004767  055406          6$:     JSR     R7,DATVER                       ;VERIFY DATAT
12688 062472  005767  120422                  TST COUNT                       ;
12689 062476  001403                          BEQ     7$                      ;BRANC IF GOOD
12690 062500  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12691 062504  104003                          ERROR   .3                      ;FPP ERROR
12692                                                                         ;BAD AC0
12693 062506                          7$:
12694                                 ;
12697                                 ;
12698                                 ;-----------------------------------
12699                                 ;TEST ADDF,SUBD - FSRC=0, AC0 NE 0
12700                                 ;
12701 062506                          MNNRM2:
12702                                 ;
12703 062506  012701  003142                  MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
12704 062512  012705  003314                  MOV     #TAB6,R5                ;POINT TO SOURCE DATA TABLE
12705 062516  012704  003244                  MOV     #TAB2,R4                ;POINT TO AC0 DATA
12706 062522  170127  000200                  LDFPS   #200                    ;SET TO DOUBLE FOR CLEAR
12707 062526  172415                  LDD     (R5),AC0                        ;
12708 062530  005002                  CLR     R2                              ;SETUP FPP STATUS
12709 062532  170102                  LDFPS   R2                              ;LOAD FPS
12710 062534  172414                  LDF     (R4),AC0                        ;LOAD AC0
12711 062536  172015                  ADDF    (R5),AC0                        ;*TEST INSTRUCTION
12712 062540  170203                  STFPS   R3                              ;SAVE STATUS
12713 062542  174011                  STF     AC0,(R1)                        ;SAVE AC0
12714 062544  022703  000000                  CMP     #0,R3                   ;VERFIY NEGATIVE RESULT
```

FLOATING POINT TESTS

```
12715 062550  001403                    BEQ     1$                      ;BRANCH IF GOOD
12716 062552  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12717 062556  104003                    ERROR   +3                      ;FPP ERROR
12718                                                                   ;BAD FPS
12719 062560  012704  003264      1$:   MOV     #TAB4,R4                ;POINT TO EXPECTED DATA
12720 062564  004767  055310            JSR     R7,DATVER                       ;VERFIY ACO
12721 062570  005767  120324            TST     COUNT                   ;CHECK RESULT
12722 062574  001403                    BEQ     2$                      ;BRANCH IF GOOD
12723 062576  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12724 062602  104003                    ERROR   +3                      ;FPP ERROR
12725                                                                   ;BAD ACO
12726 062604  170127  000200      2$:   LDFPS   #200                    ;SET STATUS TO DUOBLE NODE
12727 062610  172414                    LDD     (R4),AC0                ;LOAD AC0 WITH A VALUE
12728 062612  173015                    SUBD    (R5),AC0                ;*TEST INSTRUCTION
12729 062614  170203                    STFPS   R3                      ;SAVE FPP STATUS
12730 062616  174011                    STD     AC0,(R1)                ;SAVE AC0
12731 062620  022703  000200            CMP     #200,R3                 ;VERIFY RESULT
12732 062624  001403                    BEQ     3$                      ;BRANCH IF GOOD
12733 062626  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12734 062632  104003                    ERROR   +3                      ;FPP ERROR
12735                                                                   ;BAD SUBD
12736 062634  012704  003264      3$:   MOV     #TAB4,R4                ;POINT TO EXPECTED
12737 062640  004767  055234            JSR     R7,DATVER               ;VERIFY AC0
12738 062644  005767  120250            TST     COUNT                   ;
12739 062650  001403                    BEQ     4$                      ;BRANCH IF GOOD AC0
12740 062652  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12741 062656  104003                    ERROR   +3                      ;FPP ERROR
12742                                                                   ;BAD AC0
12743 062660                      4$:
12746                                   ;
12747                                   ;---------------------------------
12748                                   ;TEST ADDD, SUBF - FSRC NE 0, AC0=0
12749                                   ;
12750 062660                      MNNRM3:
12751                                   ;
12752 062660  012701  003142            MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
12753 062664  012705  003314            MOV     #TAB6,R5                ;POINT TO AC0 DATA TABLE
12754 062670  012704  003234            MOV     #TAB1,R4                ;POINT TO FSRC DATA
12755 062674  012702  000200            MOV     #200,R2                 ;SETUP FPP STATUS
12756 062700  170102                    LDFPS   R2                      ;LOAD FPS
12757 062702  172415                    LDD     (R5),AC0                ;LOAD AC0
12758 062704  172014                    ADDD    (R4),AC0                        ;*TEST INSTRUCTION
12759 062706  170203                    STFPS   R3                      ;SAVE STATUS
12760 062710  174011                    STD     AC0,(R1)                ;SAVE AC0
12761 062712  022703  000210            CMP     #210,R3                 ;VERFIY NEGATIVE RESULT
12762 062716  001403                    BEQ     1$                      ;BRANCH IF GOOD
12763 062720  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12764 062724  104003                    ERROR   +3                      ;FPP ERROR
12765                                                                   ;BAD FPS
12766 062726  004767  055146      1$:   JSR     R7,DATVER                       ;VERFIY AC0
12767 062732  005767  120162            TST     COUNT                   ;CHECK RESULT
12768 062736  001403                    BEQ     2$                      ;BRANCH IF GOOD
12769 062740  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12770 062744  104003                    ERROR   +3                      ;FPP ERROR
12771                                                                   ;BAD AC0
12772 062746  170127  000200      2$:   LDFPS   #200                    ;SET STATUS TO DUOBLE NODE
12773 062752  172415                    LDF     (R5),AC0                ;LOAD AC0 WITH A VALUE
```

FLOATING POINT TESTS

```
12774 062754  173014                    SUBD    (R4),AC0                ;*TEST INSTRUCTION
12775 062756  170203                    STFPS   R3                      ;SAVE FPP STATUS
12776 062760  174011                    STF     AC0,(R1)                        ;SAVE AC0
12777 062762  022703  000200            CMP     #200,R3         ;VERIFY RESULT
12778 062766  001403                    BEQ     3$                      ;BRANCH IF GOOD
12779 062770  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12780 062774  104003                    ERROR   +3                      ;FPP ERROR
12781                                                                   ;BAD SUBD
12782 062776  012704  003504      3$:   MOV     #TAB18,R4               ;POINT TO EXPECTED DATA
12783 063002  004767  055072            JSR     R7,DATVER               ;VERIFY AC0
12784 063006  005767  120106            TST     COUNT                   ;
12785 063012  001403                    BEQ     4$                      ;BRANCH IF GOOD AC0
12786 063014  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12787 063020  104003                    ERROR   +3                      ;FPP ERROR
12788                                                                   ;BAD AC0
12789 063022                      4$:
12790                                   ;
12793                                   ;
12794                                   ;---------------------------------
12795                                   ;TEST  ADDF, SUBD - EXP(AC0) = EXP(FSRC)
12796                                   ;
12797 063022                      MNNRM4:
12798                                   ;
12799 063022  012702  003240                    MOV     #3240,R2                ;SET FIU,FD,FT
12800 063026  170102                    LDFPS   R2
12801 063030  012704  003334                    MOV     #TAB7,R4                        ;SET FSRC
12802 063034  012705  003344                    MOV     #TAB8,R5                ;SETUP AC0
12803 063040  012701  003142                    MOV     #RECDST,R1              ;POINT TO RECEIVED DATA
12804 063044  172415                            LDD     (R5),AC0                        ;LOAD AC0
12805 063046  172014                            ADDD    (R4),AC0                ;*TEST INSTRUCTION
12806 063050  174011                            STD     AC0,(R1)                ;SAVE TEST RESULT
12807 063052  012704  003354                    MOV     #TAB9,R4                ;POINT TO EXPECTED DATA
12808 063056  004767  055016                    JSR     R7,DATVER                       ;VERIFY AC0 DATA
12809 063062  005767  120032                    TST     COUNT                   ;
12810 063066  001403                            BEQ     1$                      ;BRANCH IF GOOD
12811 063070  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12812 063074  104003                            ERROR   +3              ;FPP ERROR
12813                                                                   ;BAD ADDD
12814 063076  012704  003344      1$:   MOV     #TAB8,R4                        ;
12815 063102  012703  003344            MOV     #TAB8,R3                ;SETUP SAME AC0
12816 063106  012702  003200            MOV     #3200,R2                ;ROUND MODE
12817 063112  170102                    LDFPS   R2
12818 063114  172413                    LDD     (R3),AC0                ;LOAD AC0
12819 063116  061400                    ADD     (R4),AC0                ;*TEST INSTRUCTION
12820 063120  174011                    STD     AC0,(R1)                ;SAVE DATA
12821 063122  004767  054752            JSR     R7,DATVER               ;VERIFY AC0
12822 063126  005767  117766            TST     COUNT
12823 063132  001403                    BEQ     2$                      ;BRANCH IF GOOD
12824 063134  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12825 063140  104003                    ERROR   +3                      ;FPP ERROR
12826                                                                   ;BAD ROUND RESULT
12827 063142                      2$:
12828                                   ;
12831                                   ;
12832                                   ;---------------------------------
12833                                   ;TEST ADD - EXP(FSRC) > EXP(AC0)
12834                                   ;
```

FLOATING POINT TESTS

```
12835 063142                          MXDF1:
12836                                 ;
12837 063142  012702  003200                  MOV     #3200,R2                ;R2=FPP STATUS
12838 063146  170102                          LDFPS   R2                      ;LOAD FPS STATUS
12839 063150  012704  003374                  MOV     #TAB11,R4               ;POINT TO FSRC DATA
12840 063154  012701  003142                  MOV     #RECDST,R1              ;POINT TO AC0 RESULT
12841 063160  012705  003364                  MOV     #TAB10,R5               ;POINT TO AC0 DATA
12842 063164  172415                          LDD     (R5),AC0                ;LOAD AC0 DATA
12843 063166  172014                          ADDD    (R4),AC0                ;*TEST INSTRUCTIONS
12844 063170  170203                          STFPS   R3                      ;SAVE FPP STATUS
12845 063172  174011                          STD     AC0,(R1)                ;SAVE AC0 DATA
12846 063174  022703  003200                  CMP     #3200,R3                ;VERIFY FPP STATUS
12847 063200  001403                          BEQ     1$                      ;BRANCH IF GOOD
12848 063202  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12849 063206  104003                          ERROR   +3              ;FPP ERROR
12850                                         ;BAD FPP STATUS
12851 063210  012704  003404          1$:     MOV     #TAB11A,R4              ;POINT TO EXPECTED DATA
12852 063214  004767  054660                  JSR     R7,DATVER               ;VERIFY CONTENTS OF AC0
12853 063220  005767  117674                  TST     COUNT                   ;
12854 063224  001403                          BEQ     2$                      ;BRANCH IF GOOD AC0
12855 063226  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12856 063232  104003                          ERROR   +3              ;FPP ERROR
12857                                         ;BAD AC0, SHOULD = FSRC
12858 063234  012704  003414          2$:     MOV     #TAB12,R4               ;POINT TO FSRC DATA
12859 063240  172415                          LDD     (R5),AC0                ;AC0
12860 063242  172014                          ADDD    (R4),AC0                ;*TEST INSTRUCTION
12861 063244  012704  003434                  MOV     #TAB13B,R4              ;POINT TO EXPECTED RESULT
12862 063250  174011                          STD     AC0,(R1)                ;SAVE AC0 DATA INTO RECDAT
12863 063252  004767  054622                  JSR     R7,DATVER               ;VERIFY DATA
12864 063256  005767  117636                  TST     COUNT
12865 063262  001403                          BEQ     3$                      ;BRANCH IF GOOD DATA
12866 063264  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12867 063270  104003                          ERROR   +3              ;FPP ERROR
12868                                         ;BAD AC0 DATA
12869 063272  012702  003000          3$:     MOV     #3000,R2                ;GET FPP STATUS DATA
12870 063276  012704  003444                  MOV     #TAB14,R4               ;POINT TO FSRC DATA
12871 063302  012705  003454                  MOV     #TAB15,R5               ;POINT TO AC0 DATA
12872 063306  172415                          LDD     (R5),AC0                ;LOAD AC0
12873 063310  170102                          LDFPS   R2                      ;FPP STATUS = FLOAT, INTERRUPTS ENABLE
12874 063312  172014                          ADDF    (R4),AC0                ;*TEST INSTRUCTION
12875 063314  170127  000200                  LDFPS   #200                    ;RESET TO DOUBLE
12876 063320  174011                          STD     AC0, (R1)               ;RECDST=AC0
12877 063322  012704  003364                  MOV     #TAB10,R4               ;POINT TO GOOD DATA
12878 063326  004767  054546                  JSR     R7,DATVER                       ;VERFIY CONTENTS OF AC0
12879 063332  005767  117562                  TST     COUNT                   ;
12880 063336  001403                          BEQ     4$                      ;BRANCH IF GOOD
12881 063340  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12882 063344  104003                          ERROR   +3              ;FPP ERROR
12883                                         ;BAD FLOATING ADD
12884 063346  012705  003464          4$:     MOV     #TAB16,R5               ;POINT TO AC0 DATA
12885 063352  170102                          LDFPS   R2                      ;FPP STATUS = FLOAT
12886 063354  172415                          LDF     (R5),AC0                ;LOAD AC0
12887 063356  172014                          ADDF    (R4),AC0                ;*TEST INSTRUCTION
12888 063360  174011                          STD     AC0,(R1)                        ;SAVE AC0  DATA
12889 063362  012704  003474                  MOV     #TAB17,R4               ;POINT TO GOOD DATA
12890 063366  004767  054506                  JSR     R7,DATVER
12891 063372  005767  117522                  TST     COUNT
```

FLOATING POINT TESTS

```
12892 063376  001403                    BEQ     5$                      ;BRANCH IF GOOD
12893 063400  004737  140132            CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12894 063404  104003                    ERROR   +3                  ;FPP ERROR
12895                                                               ;BAD FLOATING ADD
12896 063406                      5$:
12897                             ;
12900                             ;
12901                             ;---------------------------------
12902                             ;TEST ADDD WITH NEGATIVE OPERANDS
12903                             ;
12904 063406                      MNGOP:
12905                             ;
12906 063406  012702  003200              MOV     #3200,R2                ;LOAD FPS VALUE
12907 063412  170102                      LDFPS   R2                      ;
12908 063414  012704  003514              MOV     #TAB21,R4               ;DATA ADDRESS FOR AC0 AND FSR
12909 063420  172414                      LDD     (R4),AC0                ;AC0=100200 0 0 0
12910 063422  172014                      ADDD    (R4),AC0                ;*TEST INSTRUCTION
12911 063424  170203                      STFPS   R3                      ;SAVE STATUS
12912 063426  012701  003142              MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
12913 063432  174011                      STD     AC0,(R1)                ;SAVE AC0 DATA
12914 063434  022703  003210              CMP     #3210,R3                ;VERIFY STATUS
12915 063440  001403                      BEQ     1$                      ;BRANCH IF GOOD
12916 063442  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12917 063446  104003                      ERROR   +3                  ;FPP ERROR
12918                                                               ;
12919 063450  012704  003524        1$:   MOV     #TAB22,R4               ;POINT TO EXPECTED DATA
12920 063454  004767  054420              JSR     R7,DATVER               ;
12921 063460  005767  117434              TST     COUNT                   ;VERIFY DATA
12922 063464  001403                      BEQ     2$                      ;BRANCH IF GOOD
12923 063466  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12924 063472  104003                      ERROR   +3                  ;FPP ERROR
12925                                                               ;
12926                                     ;       !-FSRC! = !AC0!
12927 063474  012704  003514        2$:   MOV     #TAB21,R4               ;POINT TO FSRC DATA
12928 063500  012701  003534              MOV     #TAB23,R1               ;POINT TO AC0 DATA
12929 063504  012737  063526  000244      MOV     #101$,@#FPVEC           ;SETUP FP VECTOR
12930 063512  172411                      LDD     (R1),AC0                ;LOAD AC0
12931 063514  172014                      ADDD    (R4),AC0                ;*TEST INSTRUCTION
12932 063516  170000              100$:   CFCC                            ;COPY FPP CC
12933 063520  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12934 063524  104003                      ERROR   +3                  ;FPP ERROR
12935                                                               ;GO TO ERROR
12936 063526  170203              101$:   STFPS   R3                      ;SAVE FPP STATUS
12937 063530  012701  003142              MOV     #RECDST,R1                      ;POINT TO RECEIVED DATA TABLE
12938 063534  174011                      STD     AC0,(R1)                ;SAVE AC0 DATA
12939 063536  022703  103200              CMP     #103200,R3              ;VERIFY STATUS
12940 063542  001403                      BEQ     3$                      ;BRANCH IF GOOD
12941 063544  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12942 063550  104003                      ERROR   +3                  ;FPP ERROR
12943                                                               ;BAD STATUS
12944 063552  012605              3$:   MOV     (SP)+,R5                ;GET ERROR PC
12945 063554  020527  063516              CMP     R5,#100$                ;VERIFY ERROR ADDRESS ON STACK
12946 063560  001403                      BEQ     102$                    ;BRANCH IF GOOD
12947 063562  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12948 063566  104003                      ERROR   +3                  ;FPP ERROR
12949                                                               ;BAD ERROR RETURN ON STACK
12950 063570  005726              102$:   TST     (SP)+                   ;RESTORE STACK
```

J2

FLOATING POINT TESTS

```
12951 063572  012704  003544                MOV     #TAB24,R4               ;POINT TO EXPECTED DATA TABLE
12952 063576  004767  054276                JSR     R7,DATVER               ;VERIFY DATA
12953 063602  005767  117312                TST     COUNT
12954 063606  001403                        BEQ     4$                      ;BRANC IF GOOD
12955 063610  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12956 063614  104003                        ERROR   +3                      ;FPP ERROR
12957                                                                       ;BAD ACO DATA
12958                                       ;       !-ACO! = !FSRC!
12959 063616  012704  003534          4$:   MOV     #TAB23,R4               ;POINT TO FSRC DATA
12960 063622  012701  003514                MOV     #TAB21,R1               ;POINT TO ACO DATA
12961 063626  012737  063656  000244        MOV     #104$,@#FPVEC           ;SETUP FP VECTOR
12962 063634  012702  003200                MOV     #3200,R2                ;LOAD FPS VALUE
12963 063640  170102                        LDFPS   R2                      ;
12964 063642  172411                        LDD     (R1),AC0                ;LOAD ACO DATA
12965 063644  172014                        ADDD    (R4),AC0                ;*TEST INSTRUCTION
12966 063646  170000                103$:   CFCC                            ;COPY FPP CC
12967 063650  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12968 063654  104003                        ERROR   +3                      ;FPP ERROR
12969                                                                       ;GO TO ERROR
12970 063656  170203                104$:   STFPS   R3                      ;SAVE FPS
12971 063660  012701  003142                MOV     #RECDST,R1              ;SAVE ACO
12972 063664  174011                        STD     AC0,(R1)                ;
12973 063666  022703  103200                CMP     #103200,R3              ;VERFIY STATUS
12974 063672  001403                        BEQ     5$                      ;BRANCH IF GOOD
12975 063674  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12976 063700  104003                        ERROR   +3                      ;FPP ERROR
12977                                                                       ;BAD FPS STATUS
12978 063702  012605                5$:     MOV     (SP)+,R5                ;GET ERROR PC
12979 063704  020527  063646                CMP     R5,#103$                ;VERIFY ERROR ADDRESS ON STACK
12980 063710  001403                        BEQ     105$                    ;BRANCH IF GOOD
12981 063712  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12982 063716  104003                        ERROR   +3                      ;FPP ERROR
12983                                                                       ;BAD ERROR RETURN ON STACK
12984 063720  005726                105$:   TST     (SP)+                   ;RESTORE STACK
12985 063722  012704  003544                MOV     #TAB24,R4               ;POINT TO EXPECTED DATA
12986 063726  004767  054146                JSR     R7,DATVER
12987 063732  005767  117162                TST     COUNT                   ;
12988 063736  001403                        BEQ     6$                      ;BRANCH IF  GOOD
12989 063740  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
12990 063744  104003                        ERROR   +3                      ;FPP ERROR
12991                                                                       ;BAD ACO
12992                                       ;       !-FSRC! <               !ACO!
12993 063746  012704  003564          6$:   MOV     #TAB26,R4               ;POINT TO FSRC DATA
12994 063752  012701  003554                MOV     #TAB25,R1               ;POINT TO ACO DATA
12995 063756  012702  003200                MOV     #3200,R2                ;LOAD FPS VALUE
12996 063762  170102                        LDFPS   R2                      ;
12997 063764  012737  000246  000244        MOV     #246,@#FPVEC            ;SETUP FP VECTOR
12998 063772  172411                        LDD     (R1),AC0                ;LOAD ACO DATA
12999 063774  172014                        ADDD    (R4),AC0                ;*TEST INSTRUCTION
13000 063776  170203                        STFPS   R3                      ;SAVE STATUS
13001 064000  012701  003142                MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13002 064004  174011                        STD     AC0,(R1)                        ;SAVE ACO
13003 064006  020327  003200                CMP     R3,#3200                        ;VERIFY STATUS
13004 064012  001403                        BEQ     7$                      ;BRANCH IF GOOD
13005 064014  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13006 064020  104003                        ERROR   +3                      ;FPP ERROR
13007                                                                       ;BAD FPS
```

FLOATING POINT TESTS

```
13008 064022  012704  003574          7$:     MOV     #TAB27,R4               ;POINT TO EXPECTED DSATA
13009 064026  004767  054046                  JSR     R7,DATVER               ;VERIFY DATA
13010 064032  005767  117062                  TST     COUNT                   ;
13011 064036  001403                          BEQ     8$                      ;BRANCH IF GOOD
13012 064040  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13013 064044  104003                          ERROR   ;3                      ;FPP ERROR
13014                                                                         ;
13015                                         ;       !FSRC!  >               !-AC!
13016 064046  012704  003554          8$:     MOV     #TAB25,R4               ;POINT TO FSRC DATA
13017 064052  012701  003564                  MOV     #TAB26,R1               ;POINT TO ACO DATA
13018 064056  172411                          LDD     (R1),AC0                ;LOAD ACO DATA
13019 064060  172014                          ADDD    (R4),AC0                ;*TEST INSTRUCTION
13020 064062  170203                          STFPS   R3                      ;SAVE STATUS
13021 064064  012701  003142                  MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13022 064070  174011                          STD     AC0,(R1)                ;SAVE ACO
13023 064072  020327  003200                  CMP     R3,#3200                ;VERIFY STATUS
13024 064076  001403                          BEQ     9$                      ;BRANCH IF GOOD
13025 064100  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13026 064104  104003                          ERROR   ;3                      ;FPP ERROR
13027                                                                         ;BAD FPS
13028 064106  012704  003574          9$:     MOV     #TAB27,R4               ;POINT TO EXPECTED DSATA
13029 064112  004767  053762                  JSR     R7,DATVER               ;VERIFY DATA
13030 064116  005767  116776                  TST     COUNT                   ;
13031 064122  001403                          BEQ     10$                     ;BRANCH IF GOOD
13032 064124  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13033 064130  104003                          ERROR   ;3                      ;FPP ERROR
13034                                                                         ;BAD ACO
13035                                         ;       !-FSRC! <               !ACO!
13036 064132  012704  003614          10$:    MOV     #TAB29,R4               ;POINT TO FSRC DATA
13037 064136  012701  003604                  MOV     #TAB28,R1               ;POINT TO ACO DATA
13038 064142  172411                          LDD     (R1),AC0                ;LOAD ACO DATA
13039 064144  172014                          ADDD    (R4),AC0                ;*TEST INSTRUCTION
13040 064146  170203                          STFPS   R3                      ;SAVE STATUS
13041 064150  012701  003142                  MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13042 064154  174011                          STD     AC0,(R1)                ;SAVE ACO
13043 064156  020327  003200                  CMP     R3,#3200                ;VERIFY STATUS
13044 064162  001403                          BEQ     11$                     ;BRANCH IF GOOD
13045 064164  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13046 064170  104003                          ERROR   ;3                      ;FPP ERROR
13047                                                                         ;BAD FPS
13048 064172  012704  003624          11$:    MOV     #TAB29A,R4              ;POINT TO EXPECTED DATA
13049 064176  004767  053676                  JSR     R7,DATVER               ;VERIFY DATA
13050 064202  005767  116712                  TST     COUNT                   ;
13051 064206  001403                          BEQ     12$                     ;BRANCH IF GOOD
13052 064210  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13053 064214  104003                          ERROR   ;3                      ;FPP ERROR
13054                                                                         ;
13055 064216                          12$:
13056                                 ;
13059                                 ;
13060                                 ;---------------------------------
13061                                 ;TEST SUB WITH EXP[AC0]=EXP[FSRC]
13062                                 ;
13063 064216                          MSB:
13064                                 ;
13065 064216  012702  003200                  MOV     #3200,R2                ;LOAD FPS DATA
13066 064222  170102                          LDFPS   R2                      ;LOAD FPS
```

FLOATING POINT TESTS

```
13067 064224  012704  003514              MOV     #TAB21,R4               ;POINT TO FSRC DATA
13068 064230  012701  003142              MOV     #RECDST,R1              ;POINT TO ACO RECEIVED DATA TABLE
13069 064234  172414                      LDD     (R4),AC0                ;LOAD ACO
13070 064236  173014                      SUBD    (R4),AC0                        ;*TEST INSTRUCTION
13071 064240  170203                      STFPS   R3                      ;SAVE STATUS
13072 064242  174011                      STD     AC0,(R1)                ;SAVE ACO INTO RECDST
13073 064244  022703  003204              CMP     #3204,R3                ;VERIFY STATUS
13074 064250  001403                      BEQ     1$                      ;BRANCH IF GOOD
13075 064252  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13076 064256  104003                      ERROR   .+3                 ;FPP ERROR
13077                                                                 ;BAD FPS STATUS
13078 064260  012704  003314        1$:   MOV     #TAB6,R4                ;POINT TO EXPECTED DATA
13079 064264  004767  053610              JSR     R7,DATVER               ;VERIFY ACO
13080 064270  005767  116624              TST     COUNT                   ;
13081 064274  001403                      BEQ     2$                      ;BRANCH IF GOOD
13082 064276  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13083 064302  104003                      ERROR   .+3                 ;FPP ERROR
13084                                                                 ;BAD ACO
13085 064304  012704  003444        2$:   MOV     #TAB14,R4               ;POINT TO FSRC AND ACO DATA
13086 064310  172414                      LDD     (R4),AC0                        ;LOAD ACO DATA
13087 064312  173014                      SUBD    (R4),AC0                        ;*TEST  INSTRUCTION
13088 064314  170203                      STFPS   R3                      ;SAVE FPS
13089 064316  174011                      STD     AC0,(R1)                        ;SAVE ACO INTO RECDST
13090 064320  022703  003204              CMP     #3204,R3                        ;VERIFY FPS
13091 064324  001403                      BEQ     3$                      ;BRANCH IF GOOD
13092 064326  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13093 064332  104003                      ERROR   .+3                 ;FPP ERROR
13094                                                                 ;BAD ACO
13095 064334  012704  003314        3$:   MOV     #TAB6,R4                ;POINT TO EXPECTED DATA
13096 064340  004767  053534              JSR     R7,DATVER                       ;VERIFY ACO
13097 064344  005767  116550              TST     COUNT
13098 064350  001403                      BEQ     4$                      ;BRANCH IF GOOD
13099 064352  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13100 064356  104003                      ERROR   .+3                 ;FPP ERROR
13101                                                                 ;BAD ACO
13102 064360                        4$:
13103                               ;
13106                               ;
13107                               ;---------------------------------
13108                               ;TEST NORMALIZE
13109                               ;
13110 064360                        MNRM:
13111                               ;
13112 064360  012702  003200              MOV     #3200,R2                ;LOAD FPS
13113 064364  170102                      LDFPS   R2                      ;
13114 064366  012705  003644              MOV     #TAB31,R5               ;POINT TO FSRC DATA
13115 064372  012701  003634              MOV     #TAB30,R1               ;POINT TO ACO DATA
13116 064376  172411                      LDD     (R1),AC0                ;LOAD ACO
13117 064400  173015                      SUBD    (R5),AC0                ;*TEST INSTRUCTION
13118                                                                     ;1 LEFT SHIFT
13119 064402  170203                      STFPS   R3                      ;SAVE STATUS
13120 064404  012704  003142              MOV     #RECDST,R4              ;POINT TO RECDATA
13121 064410  174014                      STD     AC0,(R4)                        ;SAVE ACO
13122 064412  012701  003674              MOV     #TAB34,R1               ;POINT TO EXPECTED DATA
13123 064416  004767  053456              JSR     R7,DATVER                       ;VERIFY DATA
13124 064422  005767  116472              TST     COUNT
13125 064426  001403                      BEQ     1$                      ;BRANCH IF GOOD
```

FLOATING POINT TESTS

```
13126 064430  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13127 064434  104003                        ERROR   .+3                     ;FPP ERROR
13128                                                                       ;
13129 064436  012701  003654          1$:   MOV     #TAB32,R1               ;ACO DATA
13130 064442  012705  003664                MOV     #TAB33,R5                       ;FSRC DATA
13131 064446  172411                        LDD     (R1),AC0                        ;LOAD AC0
13132 064450  173015                        SUBD    (R5),AC0                ;*TST INSTRUCTION
13133                                                                       ;56 LEFT SHIFTS
13134 064452  012701  003142                MOV     #RECDST,R1                      ;SAVE DATA
13135 064456  174011                        STD     AC0,(R1)                        ;
13136 064460  004767  053414                JSR     R7,DATVER
13137 064464  005767  116430                TST     COUNT
13138 064470  001403                        BEQ     2$                      ;
13139 064472  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13140 064476  104003                        ERROR   .+3                     ;FPP ERROR
13141                                                                       ;
13142 064500                          2$:
13143                                       ;
13146                                       ;
13147                                       ;---------------------------------
13148                                       ;TEST ADDD WITH OVERFLOW AND UNDERFLOW
13149                                       ;
13150 064500                          MUVAD:
13151                                       ;
13152 064500  012702  000200                MOV     #200,R2                 ;SETUP FLOATING POINT STATUS
13153 064504  170102                        LDFPS   R2                      ;LOAD FPS
13154 064506  012704  003704                MOV     #TAB40,R4                       ;POINT TO FSRC DATA
13155 064512  012701  003704                MOV     #TAB40,R1                       ;POINT TO ACO DATA
13156 064516  172411                        LDD     (R1),AC0                        ;LOAD AC0 WITH TEST DATA
13157 064520  172014                        ADDD    (R4),AC0                ;*TEST INSTRUCTION
13158 064522  170203                        STFPS   R3                      ;SAVE FPS
13159 064524  012701  003142                MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13160 064530  174011                        STD     AC0,(R1)                ;SAVE AC0  RESULT
13161 064532  022703  000206                CMP     #206,R3                 ;VERIFY STATUS
13162 064536  001403                        BEQ     1$                      ;BRANCH IF GOOD
13163 064540  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13164 064544  104003                        ERROR   .+3                     ;FPP ERROR
13165                                                                       ;BAD FPS
13166 064546  012704  003314          1$:   MOV     #TAB6,R4                        ;POINT TO EXPECTED DATA
13167 064552  004767  053322                JSR     R7,DATVER                       ;VERIFY DATA
13168 064556  005767  116336                TST     COUNT
13169 064562  001403                        BEQ     2$                      ;BRANCH IF GOOD
13170 064564  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13171 064570  104003                        ERROR   .+3                     ;FPP ERROR
13172                                                                       ;BAD AC0
13173                                 ;OVERFLOW TRAPS ENABLED
13174                                 ;
13175 064572  012702  001200          2$:   MOV     #1200,R2                        ;SETUP FLOATING POINT STATUS
13176 064576  170102                        LDFPS   R2                      ;LOAD FPS
13177 064600  012704  003704                MOV     #TAB40,R4                       ;POINT TO FSRC DATA
13178 064604  012701  003704                MOV     #TAB40,R1                       ;POINT TO ACO DATA
13179 064610  172411                        LDD     (R1),AC0                        ;LOAD AC0 WITH TEST DATA
13180 064612  012737  064632  000244        MOV     #3$,@#FPVEC             ;CHANGE TRAP VECTOR
13181 064620  172014                        ADDD    (R4),AC0                ;*TEST INSTRUCTION
13182 064622  170000                23$:   CFCC
13183 064624  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13184 064630  104003                        ERROR   .+3                     ;FPP ERROR
```

FLOATING POINT TESTS

```
13185
13186 064632  170203                    3$:     STFPS   R3              ;FAILED TO TRAP ON OVERFLOW
13187 064634  012701  003142                    MOV     #RECDST,R1              ;SAVE FPS
13188 064640  174011                    STD     AC0,(R1)                ;POINT TO RECEIVED DATA TABLE
13189 064642  022703  101206                    CMP     #101206,R3              ;SAVE AC0  RESULT
13190 064646  001403                    BEQ     4$                              ;VERIFY STATUS
13191 064650  004737  140132                    CALL    @#DETFPA                ;BRANCH IF GOOD
13192 064654  104003                    ERROR   .3                      ;DETERMINE FLOATING POINT FAULT. $$$
13193                                                                   ;FPP ERROR
13194 064656  012600                    4$:     MOV     (SP)+,R0        ;BAD FPS
13195 064660  022700  064622                    CMP     #23$,R0                 ;CHECK STORED PC
13196 064664  001403                    BEQ     5$                              ;
13197 064666  004737  140132                    CALL    @#DETFPA                ;BRANCH IF RETURN ADDRESS IS GOOD
13198 064672  104003                    ERROR   .3                      ;DETERMINE FLOATING POINT FAULT. $$$
13199                                                                   ;FPP ERROR
13200 064674  012600                    5$:     MOV     (SP)+,R0        ;BAD RETURN ADDRESS
13201 064676  012704  003314                    MOV     #TAB6,R4                ;CLEAN UP STACK
13202 064702  004767  053172                    JSR     R7,DATVER                       ;POINT TO EXPECTED DATA
13203 064706  005767  116206                    TST     COUNT                           ;VERIFY DATA
13204 064712  001403                    BEQ     7$
13205 064714  004737  140132                    CALL    @#DETFPA                ;BRANCH IF GOOD
13206 064720  104003                    ERROR   .3                      ;DETERMINE FLOATING POINT FAULT. $$$
13207                                                                   ;FPP ERROR
13208                                   ;UNDERFLOW TRAPS DISABLED       ;BAD AC0
13209                                   ;
13210 064722  012702  000200            7$:     MOV     #200,R2                 ;SETUP FLOATING POINT STATUS
13211 064726  170102                    LDFPS   R2                      ;LOAD FPS
13212 064730  012737  140044  000244              MOV     #WLDTRP,@#FPVEC                 ;REPLACE WILD TRAP VECTOR
13213 064736  012704  003334                    MOV     #TAB7,R4                        ;POINT TO FSRC DATA
13214 064742  012701  003714                    MOV     #TAB41,R1               ;POINT TO AC0 DATA
13215 064746  172411                    LDD     (R1),AC0                ;LOAD AC0 WITH TEST DATA
13216 064750  172014                    ADDD    (R4),AC0                ;*TEST INSTRUCTION
13217 064752  170203                    STFPS   R3                      ;SAVE FPS
13218 064754  012701  003142                    MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13219 064760  174011                    STD     AC0,(R1)                ;SAVE AC0  RESULT
13220 064762  022703  000204                    CMP     #204,R3                 ;VERIFY STATUS
13221 064766  001403                    BEQ     8$                              ;BRANCH IF GOOD
13222 064770  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13223 064774  104003                    ERROR   .3                      ;FPP ERROR
13224                                                                   ;BAD FPS
13225 064776  012704  003314            8$:     MOV     #TAB6,R4                        ;POINT TO EXPECTED DATA
13226 065002  004767  053072                    JSR     R7,DATVER                       ;VERIFY DATA
13227 065006  005767  116106                    TST     COUNT
13228 065012  001401                    BEQ     9$                      ;BRANCH IF GOOD
13229 065014  104003                    ERROR   .3                      ;FPP ERROR
13230                                                                   ;BAD AC0
13231                                   ;UNDERFLOW TRAPS ENABLED
13232                                   ;
13233 065016  012702  002200            9$:     MOV     #2200,R2                ;SETUP FLOATING POINT STATUS
13234 065022  170102                    LDFPS   R2                      ;LOAD FPS
13235 065024  012737  065056  000244              MOV     #11$,@#FPVEC            ;REPOSITION TRAP VECTOR
13236 065032  012704  003334                    MOV     #TAB7,R4                        ;POINT TO FSRC DATA
13237 065036  012701  003714                    MOV     #TAB41,R1               ;POINT TO AC0 DATA
13238 065042  172411                    LDD     (R1),AC0                ;LOAD AC0 WITH TEST DATA
13239 065044  172014                    ADDD    (R4),AC0                ;*TEST INSTRUCTION
13240 065046  170000            10$:    CFCC                            ;COPTY FPP CC
13241 065050  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
```

FLOATING POINT TESTS

```
13242 065054  104003                        ERROR   .3              ;FPP ERROR
13243                                                               ;FAILED TO TRAP ON UNDERFLOW
13244 065056  170203                 11$:   STFPS   R3                      ;SAVE FPS
13245 065060  012701  003142                MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13246 065064  174011                        STD     AC0,(R1)                ;SAVE AC0  RESULT
13247 065066  022703  102210                CMP     #102210,R3                      ;VERIFY STATUS
13248 065072  001403                        BEQ     12$                     ;BRANCH IF GOOD
13249 065074  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13250 065100  104003                        ERROR   .3              ;FPP ERROR
13251                                                               ;BAD FPS
13252 065102  012605                 12$:   MOV     (SP)+,R5                ;GET ERROR PC
13253 065104  020527  065046                CMP     R5,#10$                 ;VERIFY ERROR ADDRESS ON STACK
13254 065110  001403                        BEQ     13$                     ;BRANCH IF GOOD
13255 065112  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13256 065116  104003                        ERROR   .3              ;FPP ERROR
13257                                                               ;BAD ERROR RETURN ON STACK
13258 065120  005726                 13$:   TST     (SP)+                   ;RESTORE STACK
13259 065122  012704  003304                MOV     #TAB5A,R4                       ;POINT TO EXPECTED DATA
13260 065126  004767  052746                JSR     R7,DATVER                       ;VERIFY DATA
13261 065132  005767  115762                TST     COUNT
13262 065136  001403                        BEQ     14$                     ;BRANCH IF GOOD
13263 065140  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13264 065144  104003                        ERROR   .3              ;FPP ERROR
13265                                                               ;BAD ACO
13266                                ;UNDERFLOW WITH TRAPS DISBLED - NON-ZERO RESULT
13267                                ;
13268 065146  012702  000200         14$:   MOV     #200,R2                 ;SETUP FLOATING POINT STATUS
13269 065152  170102                        LDFPS   R2                      ;LOAD FPS
13270 065154  012737  140044  000244        MOV     #WLDTRP,@#FPVEC                 ;RESTORE TRAP VECTOR
13271 065162  012704  003714                MOV     #TAB41,R4               ;POINT TO FSRC DATA
13272 065166  012701  003724                MOV     #TAB42,R1               ;POINT TO AC0 DATA
13273 065172  172411                        LDD     (R1),AC0                ;LOAD AC0 WITH TEST DATA
13274 065174  172014                        ADDD    (R4),AC0                ;*TEST INSTRUCTION
13275 065176  170203                        STFPS   R3                      ;SAVE FPS
13276 065200  012701  003142                MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13277 065204  174011                        STD     AC0,(R1)                ;SAVE AC0  RESULT
13278 065206  022703  000204                CMP     #204,R3                 ;VERIFY STATUS
13279 065212  001403                        BEQ     15$                     ;BRANCH IF GOOD
13280 065214  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13281 065220  104003                        ERROR   .3              ;FPP ERROR
13282                                                               ;BAD FPS
13283 065222  012704  003314         15$:   MOV     #TAB6,R4                        ;POINT TO EXPECTED DATA
13284 065226  004767  052646                JSR     R7,DATVER                       ;VERIFY DATA
13285 065232  005767  115662                TST     COUNT
13286 065236  001403                        BEQ     16$                     ;BRANCH IF GOOD
13287 065240  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13288 065244  104003                        ERROR   .3              ;FPP ERROR
13289                                                               ;BAD AC0
13290                                ;UNERFLOW WITH TRAPS ENABLED - NON-ZERO  RESULT
13291                                ;
13292 065246  012702  102200         16$:   MOV     #102200,R2                      ;SETUP FLOATING POINT STATUS
13293 065252  170102                        LDFPS   R2                      ;LOAD FPS
13294 065254  012737  065306  000244        MOV     #18$,@#FPVEC            ;RESTORE TRAP VECTOR
13295 065262  012704  003714                MOV     #TAB41,R4                       ;POINT TO FSRC DATA
13296 065266  012701  003724                MOV     #TAB42,R1                       ;POINT TO AC0 DATA
13297 065272  172411                        LDD     (R1),AC0                        ;LOAD AC0 WITH TEST DATA
13298 065274  172014                        ADDD    (R4),AC0                ;*TEST INSTRUCTION
```

FLOATING POINT TESTS

```
13299 065276  170000                17$:    CFCC
13300 065300  004737  140132                CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
13301 065304  104003                        ERROR   .3              ;FPP ERROR
13302                                                               ;NO TRAP ON UNDERFLOW
13303 065306  170203                18$:    STFPS   R3              ;SAVE FPS
13304 065310  012701  003142                MOV     #RECDST,R1      ;POINT TO RECEIVED DATA TABLE
13305 065314  174011                        STD     AC0,(R1)        ;SAVE AC0  RESULT
13306 065316  012600                        MOV     (SP)+,R0        ;SAVE STACK CONTENTS
13307 065320  005726                        TST     (SP)+           ;CLEAN UP STACK
13308 065322  022700  065276                CMP     #17$,R0         ;VERIFY RETURN ADDRESS
13309 065326  001403                        BEQ     19$             ;BRANCH IF GOOD
13310 065330  004737  140132                CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
13311 065334  104003                        ERROR   .3              ;FPP ERROR
13312                                                               ;BAD RETURN ADDRESS
13313 065336  022703  102204          19$:  CMP     #102204,R3              ;VERIFY STATUS
13314 065342  001403                        BEQ     20$             ;BRANCH IF GOOD
13315 065344  004737  140132                CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
13316 065350  104003                        ERROR   .3              ;FPP ERROR
13317                                                               ;BAD FPS
13318 065352  012704  003734          20$:  MOV     #TAB43,R4               ;POINT TO EXPECTED DATA
13319 065356  004767  052516                JSR     R7,DATVER               ;VERIFY DATA
13320 065362  005767  115532                TST     COUNT
13321 065366  001403                        BEQ     21$             ;BRANCH IF GOOD
13322 065370  004737  140132                CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
13323 065374  104003                        ERROR   .3              ;FPP ERROR
13324                                                               ;BAD AC0
13325 065376                        21$:
13326                               ;
13329                               ;
13330                               ;---------------------------------
13331                               ;TEST LDCFD, LDCDF
13332                               ;
13333 065376                        MLDC:
13334                               ;
13335                               ;TRUNCATE
13336 065376  012702  000300                MOV     #300,R2         ;SETUP FLOATING POINT STATUS
13337 065402  170102                        LDFPS   R2              ;LOAD FPS
13338 065404  012704  003744                MOV     #TAB45,R4               ;POINT TO FSRC DATA
13339 065410  012701  003314                MOV     #TAB6,R1                ;POINT TO AC0 DATA
13340 065414  172411                        LDD     (R1),AC0                ;LOAD AC0 WITH TEST DATA
13341 065416  177424                        LDCFD   (R4)+,AC0       ;*TEST INSTRUCTION
13342 065420  012701  003142                MOV     #RECDST,R1      ;POINT TO RECEIVED DATA TABLE
13343 065424  174011                        STD     AC0,(R1)        ;SAVE AC0  RESULT
13344 065426  022704  003750                CMP     #TAB45+4,R4     ;VERIFY AUTO-INC
13345 065432  001403                        BEQ     1$              ;BRANCH IF GOOD AUTO-INC
13346 065434  004737  140132                CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
13347 065440  104003                        ERROR   .3              ;FPP ERROR
13348                                                               ;BAD AUTO-INC
13349 065442  012704  003754          1$:   MOV     #TAB46,R4               ;POINT TO EXPECTED DATA
13350 065446  004767  052426                JSR     R7,DATVER               ;VERIFY DATA
13351 065452  005767  115442                TST     COUNT
13352 065456  001403                        BEQ     2$              ;BRANCH IF GOOD
13353 065460  004737  140132                CALL    @#DETFPA        ;DETERMINE FLOATING POINT FAULT. $$$
13354 065464  104003                        ERROR   .3              ;FPP ERROR
13355                                                               ;BAD AC0
13356
13357                               ;AUTO-INC DOUBLE MODE
                                    ;
```

FLOATING POINT TESTS

```
13358 065466  005002                2$:     CLR     R2                      ;SETUP FLOATING POINT STATUS
13359 065470  170102                        LDFPS   R2                      ;LOAD FPS
13360 065472  012704  003744                MOV     #TAB45,R4                       ;POINT TO FSRC DATA
13361 065476  012701  003464                MOV     #TAB16,R1                       ;POINT TO ACO DATA
13362 065502  172411                        LDD     (R1),AC0                        ;LOAD AC0 WITH TEST DATA
13363 065504  177424                        LDCDF   (R4)+,AC0               ;*TEST INSTRUCTION
13364 065506  020427  003754                CMP     R4,#TAB45+10            ;VERIFY AUTO-INC
13365 065512  001403                        BEQ     3$                      ;BRANCH IF GOOD
13366 065514  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13367 065520  104003                        ERROR   +3                      ;FPP ERROR
13368                                                                       ;BAD AUTO-INC ON DOUBLE
13369 065522  170203                3$:     STFPS   R3                      ;SAVE FPS
13370 065524  012701  003142                MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13371 065530  174011                        STD     AC0,(R1)                ;SAVE AC0  RESULT
13372 065532  022703  000000                CMP     #0,R3                   ;VERIFY STATUS
13373 065536  001403                        BEQ     4$                      ;BRANCH IF GOOD
13374 065540  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13375 065544  104003                        ERROR   +3                      ;FPP ERROR
13376                                                                       ;BAD FPS
13377 065546  012704  004014          4$:   MOV     #TAB49,R4                       ;POINT TO EXPECTED DATA
13378 065552  004767  052322                JSR     R7,DATVER               ;VERIFY DATA
13379 065556  005767  115336                TST     COUNT
13380 065562  001403                        BEQ     5$                      ;BRANCH IF GOOD
13381 065564  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13382 065570  104003                        ERROR   +3                      ;FPP ERROR
13383                                                                       ;BAD AC0
13384                               ;LDCFD GR7
13385                               ;
13386 065572  012702  000200          5$:   MOV     #200,R2                 ;SETUP FLOATING POINT STATUS
13387 065576  170102                        LDFPS   R2                      ;LOAD FPS
13388 065600  005003                        CLR     R3
13389 065602  177427  043243                LDCFD   #5203,AC0               ;*TEST INSTRUCTION
13390 065606  005203                        INC     R3
13391 065610  005203                        INC     R3
13392 065612  005203                        INC     R3                      ;IF LDCFD WORKED, R3 SHOULD=3
13393 065614  022703  000003                CMP     #3,R3                   ;VERIFY CORRECT PROGRAM FLOW
13394 065620  001403                        BEQ     6$                      ;BRANCH IF GOOD
13395 065622  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13396 065626  104003                        ERROR   +3                      ;FPP ERROR
13397                                                                       ;BAD PROGRAM FLOW
13398                               ;NEGATIVE OPERANDS
13399                               ;
13400 065630  012702  000200          6$:   MOV     #200,R2                 ;SETUP FLOATING POINT STATUS
13401 065634  170102                        LDFPS   R2                      ;LOAD FPS
13402 065636  012704  003764                MOV     #TAB47,R4                       ;POINT TO FSRC DATA
13403 065642  012701  003744                MOV     #TAB45,R1                       ;POINT TO ACO DATA
13404 065646  172411                        LDD     (R1),AC0                        ;LOAD AC0 WITH TEST DATA
13405 065650  177414                        LDCFD   (R4),AC0                ;*TEST INSTRUCTION
13406 065652  170203                        STFPS   R3                      ;SAVE FPS
13407 065654  012701  003142                MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13408 065660  174011                        STD     AC0,(R1)                ;SAVE AC0  RESULT
13409 065662  022703  000210                CMP     #210,R3                 ;VERIFY STATUS
13410 065666  001403                        BEQ     7$                      ;BRANCH IF GOOD
13411 065670  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13412 065674  104003                        ERROR   +3                      ;FPP ERROR
13413                                                                       ;BAD FPS
13414 065676  012704  004004          7$:   MOV     #TAB48,R4                       ;POINT TO EXPECTED DATA
```

FLOATING POINT TESTS

```
13415 065702  004767  052172                JSR     R7,DATVER                       ;VERIFY DATA
13416 065706  005767  115206                TST     COUNT
13417 065712  001403                        BEQ     8$                      ;BRANCH IF GOOD
13418 065714  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13419 065720  104003                        ERROR   +3                      ;FPP ERROR
13420                                                                       ;BAD ACO
13421                             ;LOAD A ZERO
13422                             ;
13423 065722  012702  000200      8$:       MOV     #200,R2                 ;SETUP FLOATING POINT STATUS
13424 065726  170102                        LDFPS   R2                      ;LOAD FPS
13425 065730  012704  003314                MOV     #TAB6,R4                        ;POINT TO FSRC DATA
13426 065734  012701  004004                MOV     #TAB48,R1                       ;POINT TO ACO DATA
13427 065740  172411                        LDD     (R1),AC0                        ;LOAD ACO WITH TEST DATA
13428 065742  177414                        LDCFD   (R4),AC0                        ;*TEST INSTRUCTION
13429 065744  170203                        STFPS   R3                      ;SAVE FPS
13430 065746  012701  003142                MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13431 065752  174011                        STD     AC0,(R1)                ;SAVE ACO  RESULT
13432 065754  022703  000204                CMP     #204,R3                 ;VERIFY STATUS
13433 065760  001403                        BEQ     9$                      ;BRANCH IF GOOD
13434 065762  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13435 065766  104003                        ERROR   +3                      ;FPP ERROR
13436                                                                       ;BAD FPS
13437 065770  012704  003314      9$:       MOV     #TAB6,R4                        ;POINT TO EXPECTED DATA
13438 065774  004767  052100                JSR     R7,DATVER                       ;VERIFY DATA
13439 066000  005767  115114                TST     COUNT
13440 066004  001403                        BEQ     10$                     ;BRANCH IF GOOD
13441 066006  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13442 066012  104003                        ERROR   +3                      ;FPP ERROR
13443                                                                       ;BAD ACO
13444 066014                      10$:
13445                             ;
13448                             ;
13449                             ;---------------------------------
13450                             ;TEST CMPD
13451                             ;
13452 066014                      MCMPD:
13453                             ;
13454                             ;  CMPD WITH FRSC=AC0=0
13455                             ;
13456 066014  005037  003030                      CLR     @#FLAG                  ;SIGNAL THAT ACO REMAINS CONSTANT
13457 066020  004767  000152                      JSR     R7,CMPRTN               ;ROUTINE TO TEST DATA
13458 066024  000000  000000  000000              .WORD   0,0,0,0                 ;ACO AT START
13459 066034  000000  000000  000000              .WORD   0,0,0,0                 ;FSRC AT START
13460 066044  000200                            .WORD   200                     ;FPS AT START (D)
13461 066046  000204                            .WORD   204                     ;FPS AT END
13462                             ;  CMPD WITH EXP[FSRC]=0, EXP[AC0]=0
13463 066050  012737  000001  003030            MOV     #1,@#FLAG               ;SIGNAL THAT ACO WILL = 0
13464 066056  004767  000114                    JSR     R7,CMPRTN               ;ROUTINE TO TEST DATA
13465 066062  000000  000000  000000            .WORD   0,0,0,125252            ;ACO AT START
13466 066072  000100  000022  000123            .WORD   100,22,123,123          ;FSRC AT START
13467 066102  000200                            .WORD   200                     ;FPS AT START (D)
13468 066104  000204                            .WORD   204                     ;FPS AT END
13469                             ;  CMPD FSRC>EXP[AC0]=0
13470 066106  005037  003030                    CLR     @#FLAG                  ;ACO REMAINS UNCHANGED
13471 066112  004767  000060                    JSR     R7,CMPRTN               ;ROUTINE TO TEST DATA
13472 066116  000400  012346  012346            .WORD   400,12346,12346,23      ;ACO AT START
13473 066126  000200  000000  000000            .WORD   200,0,0,0               ;FSRC AT START
```

FLOATING POINT TESTS

```
13474 066136  000200                        .WORD   200                     ;FPS AT START (D)
13475 066140  000210                        .WORD   210                     ;FPS AT END
13476                               ; CMPD FSRC=AC0>0
13477 066142  004767  000030                JSR     R7,CMPRTN               ;ROUTINE TO TEST DATA
13478 066146  077777  177777  177777        .WORD   77777,-1,-1,-1          ;AC0 AT START
13479 066156  077777  177777  177777        .WORD   77777,-1,-1,-1          ;FSRC AT START
13480 066166  000200                        .WORD   200                     ;FPS AT START (D)
13481 066170  000204                        .WORD   204                     ;FPS AT END
13482 066172  000167  000126                JMP     HOP44                   ;HOP OVER SUBROUTINE
13483
13484                               ;*X*X*X
```

FLOATING POINT TESTS

```
13533                                   ;---------------------------------
13534                                   ;TEST DIVF
13535                                   ;
13536 066324                            MDIVF:
13537                                   ;
13538                                   ;1/EXP[AC]=FSRC=0
13539 066324  012737  000002  003030            MOV     #2,@#FLAG               ;NO INTERRUPT, BUT FEC
13540 066332  004767  000706                    JSR     R7,DVFSUB               ;DO TEST
13541 066336  000100  000027                    .WORD   100,27          ;AC0
13542 066342  000000  000000                    .WORD   0,0             ;FSRC
13543 066346  000100  000027                    .WORD   100,27          ;RESULT
13544 066352  040000                            .WORD   40000           ; TEST FPS
13545 066354  140000                            .WORD   140000          ;RESULT FPS
13546 066356  000004                            .WORD   4               ;FEC
13547                                   ;2/AC=EXP[FSRC]=0
13548                                   ;TRAPS ENABLED
13549 066360  012737  000001  003030            MOV     #1,@#FLAG               ;INTERRUPT
13550 066366  004767  000652                    JSR     R7,DVFSUB               ;DO TEST
13551 066372  000000  000000                    .WORD   0,0             ;AC0
13552 066376  000100  000000                    .WORD   100,0           ;FSRC
13553 066402  000000  000000                    .WORD   0,0             ;RESULT
13554 066406  000000                            .WORD   0               ; TEST FPS
13555 066410  100000                            .WORD   100000          ;RESULT FPS
13556 066412  000004                            .WORD   4               ;FEC
13557                                   ;3/FSRC>AC0=0
13558 066414  005037  003030                    CLR     @#FLAG          ;NO INTERRUPT
13559 066420  004767  000620                    JSR     R7,DVFSUB               ;DO TEST
13560 066424  000177  000234                    .WORD   177,234         ;AC0
13561 066430  004100  000000                    .WORD   4100,0          ;FSRC
13562 066434  000000  000000                    .WORD   0,0             ;RESULT
13563 066440  007400                            .WORD   7400            ; TEST FPS
13564 066442  007404                            .WORD   7404            ;RESULT FPS
13565                                   ;4/AC0>EXP[FSRC]=0
13566 066444  012737  000001  003030            MOV     #1,@#FLAG               ;INTERRUPT
13567 066452  004767  000566                    JSR     R7,DVFSUB               ;DO TEST
13568 066456  040200  104210                    .WORD   40200,104210    ;AC0
13569 066462  000125  025252                    .WORD   125,25252       ;FSRC
13570 066466  040200  104210                    .WORD   40200,104210    ;RESULT
13571 066472  007557                            .WORD   7557            ; TEST FPS
13572 066474  107557                            .WORD   107557          ;RESULT FPS
13573 066476  000004                            .WORD   4               ;FEC
13574                                   ;5/EXP[AC]=EXP[FSRC]
13575 066500  005037  003030                    CLR     @#FLAG          ;NO INTERRUPT
13576 066504  004767  000534                    JSR     R7,DVFSUB               ;DO TEST
13577 066510  077760  177777                    .WORD   77760,-1                ;AC0
13578 066514  077760  000000                    .WORD   77760,0         ;FSRC
13579 066520  040200  104210                    .WORD   40200,104210    ;RESULT
13580 066524  007414                            .WORD   7414            ; TEST FPS
13581 066526  007400                            .WORD   7400            ;RESULT FPS
13582                                   ;6/AC=FSRC
13583 066530  005037  003030                    CLR     @#FLAG          ;NO INTERRUPT
13584 066534  004767  000504                    JSR     R7,DVFSUB               ;DO TEST
13585 066540  052525  052525                    .WORD   52525,52525             ;AC0
13586 066544  052525  052525                    .WORD   52525,52525             ;FSRC
13587 066550  040200  000000                    .WORD   40200,0         ;RESULT
13588 066554  007400                            .WORD   7400            ; TEST FPS
13589 066556  007400                            .WORD   7400            ;RESULT FPS
```

FLOATING POINT TESTS

```
13590
13591 066560  005037  003030                ;7/FSRC>0<AC0, ROUND
                                            CLR     @#FLAG          ;NO INTERRUPT
13592 066564  004767  000454                JSR     R7,DVFSUB               ;DO TEST
13593 066570  077777  125252                .WORD   77777,125252    ;AC0
13594 066574  040300  000000                .WORD   40300,0         ;FSRC
13595 066600  077652  070707                .WORD   77652,070707            ;RESULT
13596 066604  007400                        .WORD   7400            ; TEST FPS
13597 066606  007400                        .WORD   7400            ;RESULT FPS
13598                                       ;8/AC>0<FSRC
13599 066610  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
13600 066614  004767  000424                JSR     R7,DVFSUB               ;DO TEST
13601 066620  055377  177777                .WORD   55377,-1        ;AC0
13602 066624  055300  000000                .WORD   55300,0         ;FSRC
13603 066630  040252  125252                .WORD   40252,125252    ;RESULT
13604 066634  000000                        .WORD   0               ; TEST FPS
13605 066636  000000                        .WORD   0               ;RESULT FPS
13606                                       ;9/FSRC>AC>0
13607 066640  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
13608 066644  004767  000374                JSR     R7,DVFSUB               ;DO TEST
13609 066650  064600  000001                .WORD   64600,1         ;AC0
13610 066654  066600  000000                .WORD   66600,0         ;FSRC
13611 066660  036200  000001                .WORD   36200,1         ;RESULT
13612 066664  000000                        .WORD   0               ; TEST FPS
13613 066666  000000                        .WORD   0               ;RESULT FPS
13614                                       ;10/AC>FSRC>0
13615 066670  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
13616 066674  004767  000344                JSR     R7,DVFSUB               ;DO TEST
13617 066700  012345  156024                .WORD   12345,156024    ;AC0
13618 066704  005600  000000                .WORD   05600,0         ;FSRC
13619 066710  044745  156024                .WORD   44745,156024            ;RESULT
13620 066714  000017                        .WORD   17              ; TEST FPS
13621 066716  000000                        .WORD   0               ;RESULT FPS
13622                                       ;11/FSRC<0
13623 066720  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
13624 066724  004767  000314                JSR     R7,DVFSUB               ;DO TEST
13625 066730  040422  101010                .WORD   40422,101010    ;AC0
13626 066734  140511  101010                .WORD   140511,101010   ;FSRC
13627 066740  140072  020167                .WORD   140072,20167            ;RESULT
13628 066744  000057                        .WORD   57              ; TEST FPS
13629 066746  000050                        .WORD   50              ;RESULT FPS
13630                                       ;12/AC<0
13631 066750  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
13632 066754  004767  000264                JSR     R7,DVFSUB               ;DO TEST
13633 066760  160077  000101                .WORD   160077,101      ;AC0
13634 066764  040417  177777                .WORD   40417,-1        ;FSRC
13635 066770  157651  143527                .WORD   157651,143527           ;RESULT
13636 066774  000007                        .WORD   7               ; TEST FPS
13637 066776  000010                        .WORD   10              ;RESULT FPS
13638                                       ;13/TRUNCATE TEST
13639 067000  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
13640 067004  004767  000234                JSR     R7,DVFSUB               ;DO TEST
13641 067010  060100  000177                .WORD   60100,177       ;AC0
13642 067014  040300  000000                .WORD   40300,0         ;FSRC
13643 067020  060000  000124                .WORD   60000,124               ;RESULT
13644 067024  000040                        .WORD   40              ; TEST FPS
13645 067026  000040                        .WORD   40              ;RESULT FPS
13646                                       ;14/ROUND TEST
```

FLOATING POINT TESTS

```
13647 067030  005037  003030                  CLR     @#FLAG          ;NO INTERRUPT
13648 067034  004767  000204                  JSR     R7,DVFSUB               ;DO TEST
13649 067040  060100  000177                  .WORD   60100,177       ;AC0
13650 067044  040300  000000                  .WORD   40300,0 ;FSRC
13651 067050  060000  000125                  .WORD   60000,125               ;RESULT
13652 067054  000000                          .WORD   0               ; TEST FPS
13653 067056  000000                          .WORD   0               ;RESULT FPS
13654                         ;15/OVERFLOW, INTERRUPTS ENABLED
13655 067060  012737  000001  003030          MOV     #1,@#FLAG               ;INTERRUPT
13656 067066  004767  000152                  JSR     R7,DVFSUB               ;DO TEST
13657 067072  177700  000000                  .WORD   177700,0                ;AC0
13658 067076  000200  000000                  .WORD   200,0           ;FSRC
13659 067102  137700  000000                  .WORD   137700,0                ;RESULT
13660 067106  001100                          .WORD   1100            ; TEST FPS
13661 067110  101112                          .WORD   101112          ;RESULT FPS
13662 067112  000010                          .WORD   10              ;FEC
13663                         ;16/OVERFLOW, TRAPS DISABLED
13664 067114  012737  000002  003030          MOV     #2,@#FLAG               ;NO INTERRUPT
13665 067122  004767  000116                  JSR     R7,DVFSUB               ;DO TEST
13666 067126  000200  000000                  .WORD   200,0           ;AC0
13667 067132  177700  000000                  .WORD   177700,0                ;FSRC
13668 067136  000000  000000                  .WORD   0,0             ;RESULT
13669 067142  041100                          .WORD   41100           ; TEST FPS
13670 067144  041104                          .WORD   41104           ;RESULT FPS
13671 067146  000010                          .WORD   10              ;FEC OVERFLOW
13672                         ;17/UNDERFLOW, TRAPS ENABLED, UV RESULT
13673 067150  012737  000001  003030          MOV     #1,@#FLAG               ;INTERRUPT
13674 067156  004767  000062                  JSR     R7,DVFSUB               ;DO TEST
13675 067162  100200  000000                  .WORD   100200,0                ;AC0
13676 067166  040377  177777                  .WORD   40377,-1                ;FSRC
13677 067172  100000  000001                  .WORD   100000,1                ;RESULT
13678 067176  002000                          .WORD   2000            ; TEST FPS
13679 067200  102014                          .WORD   102014          ;RESULT FPS
13680 067202  000012                          .WORD   12              ;FEC
13681                         ;18/UNDERFLOW,TRAPS ENABLED, ROUND
13682 067204  012737  000001  003030          MOV     #1,@#FLAG               ;INTERRUPT
13683 067212  004767  000026                  JSR     R7,DVFSUB               ;DO TEST
13684 067216  030325  025252                  .WORD   30325,25252             ;AC0
13685 067222  076777  023456                  .WORD   76777,23456             ;FSRC
13686 067226  071525  157716                  .WORD   71525,157716            ;RESULT
13687 067232  002537                          .WORD   2537            ; TEST FPS
13688 067234  102500                          .WORD   102500          ;RESULT FPS
13689 067236  000012                          .WORD   12              ;FEC
13690                         ;
13691 067240  000167  000242                  JMP     HOP10           ;GO TO NEXT TEST
13692                         ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
13693                         ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
13694                         ;DIVF SUBROUTINE:
13695                         ;               AC0
13696                         ;               FSRC
13697                         ;               FPS BEFORE EXECUTION
13698                         ;               FPS AFTER EXECUTION
13699                         ;               (FEC)
13700                         ;
13701                         ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
13702                         ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
13703                         ;
```

FLOATING POINT TESTS

```
13704 067244  012605                    DVFSUB: MOV     (SP)+,R5                ; RETURN ADDRESS TO USE AS POINTER
13705 067246  012737  067326  000244            MOV     #50$,@#FPVEC            ;REDIRECT TRAP VECTOR
13706 067254  012702  000200                    MOV     #200,R2                 ;SET TO DOUBLE MODE FOR LOAD
13707 067260  170102                            LDFPS   R2                      ;LOAD FPS
13708 067262  010504                            MOV     R5,R4                   ;POINT TO FSRC DATA
13709 067264  062704  000004                    ADD     #4,R4
13710 067270  172415                            LDD     (R5),AC0                ;LOAD AC0 WITH TEST DATA
13711 067272  016502  000014                    MOV     14(R5),R2               ;GET TEST FPS
13712 067276  170102                            LDFPS   R2                      ;LOAD TEST FPS
13713                                   ;
13714 067300  174414                            DIVF    (R4),AC0                ;*TEST INSTRUCTION
13715 067302  170001                    1$:     SETF                            ;WAIT FOR POSSIBLE FPA TRAP.
13716                                   ;
13717                                   ;INSTRUCTION DIDNT TRAP
13718                                   ;
13719 067304  032737  000001  003030            BIT     #1,@#FLAG                       ;VERIFY A NO TRAP CONDITION
13720 067312  001426                            BEQ     2$                      ;BRANCH IF GOOD
13721 067314  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13722 067320  104003                            ERROR   +3                      ;FPP ERROR
13723                                                                           ;INSTRUCTION SHOULD HAVE TRAPPED
13724 067322  000167  000042                    JMP     2$                      ;REJOIN CODE
13725                                   ;
13726                                   ;INSTRUCTION TRAPPED
13727 067326  032737  000001  003030    50$:    BIT     #1,@#FLAG               ;SEE IF EXPECTING A TRAP
13728 067334  001005                            BNE     51$                     ;BRANCH IF EXPECTING A TRAP
13729 067336  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13730 067342  104003                            ERROR   +3                      ;FPP ERROR
13731                                                                           ;INSTRUCTION WASNT SUPPOSE TO TRAP
13732 067344  000167  000020                    JMP     2$                      ;REJOIN CODE
13733 067350  012604                    51$:    MOV     (SP)+,R4                ;SEE IF PC = INSTRUCTION
13734 067352  005726                            TST     (SP)+                   ;CLEAN UP STACK
13735 067354  022704  067302                    CMP     #1$,R4                  ;
13736 067360  001403                            BEQ     2$                      ;BRANCH IF GOOD COMPARE
13737 067362  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13738 067366  104003                            ERROR   +3                      ;FPP ERROR
13739                                                                           ;PC WAS INCORRECT
13740                                   ;
13741                                   ;COMMON CODE FOR TRAP AND NO TRAP
13742                                   ;
13743 067370  170203                    2$:     STFPS   R3                      ;SAVE FPS
13744 067372  012702  000200                    MOV     #200,R2                 ;SET FPP TO DOUBLE
13745 067376  170102                            LDFPS   R2
13746 067400  012701  003142                    MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13747 067404  174011                            STD     AC0,(R1)                ;SAVE AC0  RESULT
13748 067406  026503  000016                    CMP     16(R5),R3               ;VERIFY STATUS
13749 067412  001403                            BEQ     3$                      ;BRANCH IF GOOD
13750 067414  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13751 067420  104003                            ERROR   +3                      ;FPP ERROR
13752                                                                           ;BAD FPS
13753 067422  010504                    3$:     MOV     R5,R4                   ;POINT TO EXPECTED DATA
13754 067424  062704  000010                    ADD     #10,R4
13755 067430  004767  050426            4$:     JSR     R7,DATVFR                       ;VERIFY DATA
13756 067434  005767  113460                    TST     COUNT
13757 067440  001403                            BEQ     5$                      ;BRANCH IF GOOD
13758 067442  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13759 067446  104003                            ERROR   +3                      ;FPP ERROR
13760                                                                           ;BAD AC0
```

FLOATING POINT TESTS

```
13761 067450  005737  003030          5$:     TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
13762 067454  001002                          BNE     6$                      ;BRANCH IF NEED TO CHECK
13763 067456  000165  000020                  JMP     20(R5)                  ;RETURN FROM TEST
13764 067462  170301                  6$:     STST    R1                      ;SAVE FEC
13765 067464  016504  000020                  MOV     20(R5),R4               ;GET FEC
13766 067470  020401                          CMP     R4,R1                   ;VERIFY FEC
13767 067472  001403                          BEQ     7$                      ;BRANCH IF GOOD
13768 067474  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13769 067500  104003                          ERROR   .3              ;FPP ERROR
13770                                                                 ;BAD FEC
13771 067502  000165  000022          7$:     JMP     22(R5)                  ;RETURN FROM TEST
13772                                 ;
13773 067506                          HOP10:
13776                                 ;
13777                                 ;---------------------------------
13778                                 ;TEST DIVD -
13779                                 ;
13780 067506                          MDIVD:
13781                                 ;
13782                                 ;1/AC=FSRC=0 TRAPS DISABLED
13783 067506  012737  000002  003030          MOV     #2,@#FLAG               ;NO INTERRUPT
13784 067514  004767  000516                  JSR     R7,DVDSUB               ;DO TEST
13785 067520  000000  000000  000000          .WORD   0,0,0,1         ;AC0
13786 067530  000100  000000  000000          .WORD   100,0,0,0       ;FSRC
13787 067540  000000  000000  000000          .WORD   0,0,0,1         ;RESULT
13788 067550  040000                          .WORD   40000           ; TEST FPS
13789 067552  140000                          .WORD   140000          ;RESULT FPS
13790 067554  000004                          .WORD   4               ;FEC
13791                                 ;2/FSRC=0, TRAPS ENABLED
13792 067556  012737  000001  003030          MOV     #1,@#FLAG               ;INTERRUPT
13793 067564  004767  000446                  JSR     R7,DVDSUB               ;DO TEST
13794 067570  000402  000000  000000          .WORD   402,0,0,0       ;AC0
13795 067600  000000  000000  000000          .WORD   0,0,0,0         ;FSRC
13796 067610  000402  000000  000000          .WORD   402,0,0,0               ;RESULT
13797 067620  000200                          .WORD   200             ; TEST FPS
13798 067622  100200                          .WORD   100200          ;RESULT FPS
13799 067624  000004                          .WORD   4                       ;FEC
13800                                 ;3/ROUND
13801 067626  005037  003030                  CLR     @#FLAG          ;NO INTERRUPT
13802 067632  004767  000400                  JSR     R7,DVDSUB               ;DO TEST
13803 067636  034300  000000  000000          .WORD   34300,0,0,1     ;AC0
13804 067646  140300  000000  000000          .WORD   140300,0,0,0    ;FSRC
13805 067656  134200  000000  000000          .WORD   134200,0,0,1    ;RESULT
13806 067666  000200                          .WORD   200             ; TEST FPS
13807 067670  000210                          .WORD   210             ;RESULT FPS
13808                                 ;4/TRUNCATE
13809 067672  005037  003030                  CLR     @#FLAG          ;NO INTERRUPT
13810 067676  004767  000334                  JSR     R7,DVDSUB               ;DO TEST
13811 067702  034300  000000  000000          .WORD   34300,0,0,1     ;AC0
13812 067712  140300  000000  000000          .WORD   140300,0,0,0    ;FSRC
13813 067722  134200  000000  000000          .WORD   134200,0,0,0    ;RESULT
13814 067732  000240                          .WORD   240             ; TEST FPS
13815 067734  000250                          .WORD   250             ;RESULT FPS
13816                                 ;5/ROUND NEGATIVE AC, FSRC
13817 067736  005037  003030                  CLR     @#FLAG          ;NO INTERRUPT
13818 067742  004767  000270                  JSR     R7,DVDSUB               ;DO TEST
13819 067746  177642  000000  000000          .WORD   177642,0,0,151  ;AC0
```

FLOATING POINT TESTS

```
13820 067756  166600  000000  000000          .WORD   166600,0,0,123  ;FSRC
13821 067766  051242  000000  000000          .WORD   51242,0,0,0             ;RESULT
13822 067776  000200                          .WORD   200             ; TEST FPS
13823 070000  000200                          .WORD   200             ;RESULT FPS
13824                               ;6/TRUNCATE NEAGTIVE AC, FSRC
13825 070002  005037  003030                  CLR     @#FLAG          ;NO INTERRUPT
13826 070006  004767  000224                  JSR     R7,DVDSUB               ;DO TEST
13827 070012  177642  000000  000000          .WORD   177642,0,0,151  ;AC0
13828 070022  166600  000000  000000          .WORD   166600,0,0,123  ;FSRC
13829 070032  051241  177777  177777          .WORD   51241,-1,-1,-1          ;RESULT
13830 070042  000240                          .WORD   240             ; TEST FPS
13831 070044  000240                          .WORD   240             ;RESULT FPS
13832                               ;7/AC=FSRC
13833 070046  005037  003030                  CLR     @#FLAG          ;NO INTERRUPT
13834 070052  004767  000160                  JSR     R7,DVDSUB               ;DO TEST
13835 070056  055521  047621  100333          .WORD   55521,47621,100333,-1   ;AC0
13836 070066  055521  047621  100333          .WORD   55521,47621,100333,-1   ;FSRC
13837 070076  040200  000000  000000          .WORD   40200,0,0,0             ;RESULT
13838 070106  007717                          .WORD   7717            ; TEST FPS
13839 070110  007700                          .WORD   7700            ;RESULT FPS
13840                               ;8/UNDERFLOW TRAPS ENABLED, UV RESULT
13841 070112  012737  000001  003030          MOV     #1,@#FLAG               ;INTERRUPT
13842 070120  004767  000112                  JSR     R7,DVDSUB               ;DO TEST
13843 070124  100200  000000  000000          .WORD   100200,0,0,0            ;AC0
13844 070134  077777  000000  000000          .WORD   77777,0,0,0             ;FSRC
13845 070144  140400  100200  100200          .WORD   140400,100200,100200,100201     ;RESULT
13846 070154  002200                          .WORD   2200            ; TEST FPS
13847 070156  102210                          .WORD   102210          ;RESULT FPS
13848 070160  000012                          .WORD   12              ;FEC
13849                               ;9/OVERFLOW TRAPS ENABLED
13850 070162  012737  000001  003030          MOV     #1,@#FLAG               ;INTERRUPT
13851 070170  004767  000042                  JSR     R7,DVDSUB               ;DO TEST
13852 070174  077000  123465  012346          .WORD   77000,123465,12346,525          ;AC0
13853 070204  000303  000001  140000          .WORD   303,1,140000,140001             ;FSRC
13854 070214  036650  163002  103645          .WORD   36650,163002,103645,64003       ;RESULT
13855 070224  001700                          .WORD   1700            ; TEST FPS
13856 070226  101702                          .WORD   101702          ;RESULT FPS
13857 070230  000010                          .WORD   10              ;FEC
13858                               ;
13859 070232  000167  000242                  JMP     HOP11                   ;HOP OVER SUBROUTINE
13860                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
13861                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
13862                               ;
13863                               ;DIVD SUBROUTINE:
13864                               ;               AC0
13865                               ;               FSRC
13866                               ;               FPS BEFORE EXECUTION
13867                               ;               FPS AFTER EXECUTION
13868                               ;               (FEC)
13869                               ;
13870                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
13871                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
13872                               ;
13873 070236  012605                  DVDSUB: MOV     (SP)+,R5                ; RETURN ADDRESS TO USE AS POINTER
13874 070240  012737  070320  000244          MOV     #50$,@#FPVEC            ;REDIRECT TRAP VECTOR
13875 070246  012702  000200                  MOV     #200,R2                 ;SET TO DOUBLE MODE FOR LOAD
13876 070252  170102                          LDFPS   R2                      ;LOAD FPS
```

FLOATING POINT TESTS

```
13877 070254  010504                        MOV     R5,R4                   ;POINT TO FSRC DATA
13878 070256  062704  000010                ADD     #10,R4
13879 070262  172415                        LDD     (R5),AC0                ;LOAD AC0 WITH TEST DATA
13880 070264  016502  000030                MOV     30(R5),R2               ;GET TEST FPS
13881 070270  170102                        LDFPS   R2                      ;LOAD TEST FPS
13882                               ;
13883 070272  174414                        DIVD    (R4),AC0                ;*TEST INSTRUCTION
13884 070274  170000                1$:     CFCC                            ;WAIT FOR POSSIBLE FPA TRAP.
13885                               ;
13886                               ;INSTRUCTION DIDNT TRAP
13887                               ;
13888 070276  032737  000001  003030        BIT     #1,@#FLAG                       ;VERIFY A NO TRAP CONDITION
13889 070304  001426                        BEQ     2$                      ;BRANCH IF GOOD
13890 070306  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13891 070312  104003                        ERROR   +3              ;FPP ERROR
13892                                                                 ;INSTRUCTION SHOULD HAVE TRAPPED
13893 070314  000167  000042                JMP     2$                      ;REJOIN CODE
13894                               ;
13895                               ;INSTRUCTION TRAPPED
13896                               ;
13897 070320  032737  000001  003030  50$:   BIT     #1,@#FLAG               ;SEE IF EXPECTING A TRAP
13898 070326  001005                        BNE     51$                     ;BRANCH IF EXPECTING A TRAP
13899 070330  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13900 070334  104003                        ERROR   +3              ;FPP ERROR
13901                                                                 ;INSTRUCTION WASNT SUPPOSE TO TRAP
13902 070336  000167  000020                JMP     2$                      ;REJOIN CODE
13903 070342  012604                51$:    MOV     (SP)+,R4                ;SEE IF PC = INSTRUCTION
13904 070344  005726                        TST     (SP)+                   ;CLEAN UP STACK
13905 070346  022704  070274                CMP     #1$,R4                  ;
13906 070352  001403                        BEQ     2$                      ;BRANCH IF GOOD COMPARE
13907 070354  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13908 070360  104003                        ERROR   +3              ;FPP ERROR
13909                                                                 ;PC WAS INCORRECT
13910                               ;
13911                               ;COMMON CODE FOR TRAP AND NO TRAP
13912                               ;
13913 070362  170203                2$:     STFPS   R3                      ;SAVE FPS
13914 070364  012702  000200                MOV     #200,R2                 ;SET FPP TO DOUBLE
13915 070370  170102                        LDFPS   R2
13916 070372  012701  003142                MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
13917 070376  174011                        STD     AC0,(R1)                ;SAVE AC0  RESULT
13918 070400  026503  000032                CMP     32(R5),R3               ;VERIFY STATUS
13919 070404  001403                        BEQ     3$                      ;BRANCH IF GOOD
13920 070406  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13921 070412  104003                        ERROR   +3              ;FPP ERROR
13922                                                                 ;BAD FPS
13923 070414  010504                3$:     MOV     R5,R4                   ;POINT TO EXPECTED DATA
13924 070416  062704  000020                ADD     #20,R4
13925 070422  004767  047452          4$:   JSR     R7,DATVER                       ;VERIFY DATA
13926 070426  005767  112466                TST     COUNT
13927 070432  001403                        BEQ     5$                      ;BRANCH IF GOOD
13928 070434  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13929 070440  104003                        ERROR   +3              ;FPP ERROR
13930                                                                 ;BAD AC0
13931 070442  005737  003030          5$:   TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
13932 070446  001002                        BNE     6$                      ;BRANCH IF NEED TO CHECK
13933 070450  000165  000034                JMP     34(R5)                  ;RETURN FROM TEST
```

FLOATING POINT TESTS

```
13934 070454  170301                  6$:     STST    R1                      ;SAVE FEC
13935 070456  016504  000034                  MOV     34(R5),R4               ;GET FEC
13936 070462  020401                          CMP     R4,R1                   ;VERIFY FEC
13937 070464  001403                          BEQ     7$                      ;BRANCH IF GOOD
13938 070466  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
13939 070472  104003                          ERROR   +3                  ;FPP ERROR
13940                                                                         ;BAD FEC
13941 070474  000165  000036            7$:     JMP     36(R5)                  ;RETURN FROM TEST
13942                                   ;
13943 070500                            HOP11:
13946                                   ;
13947                                   ;---------------------------------
13948                                   ;TEST MULF
13949                                   ;
13950 070500                            MMULF:
13951                                   ;
13952                                   ;1/ACO=FSRC=0 -INTERRUPTS DISABLED
13953 070500  005037  003030                  CLR     @#FLAG              ;NO INTERRUPT
13954 070504  004767  000564                  JSR     R7,MLFSUB                   ;DO TEST
13955 070510  000000  000000                  .WORD   0,0     ;ACO
13956 070514  000000  000000                  .WORD   0,0     ;FSRC
13957 070520  000000  000000                  .WORD   0,0                 ;RESULT
13958 070524  007517                          .WORD   7517                ; TEST FPS
13959 070526  007504                          .WORD   7504                ;RESULTANT FPS
13960                                   ;2/AC>FSRC=0 - INTERRUPTS ON
13961 070530  005037  003030                  CLR     @#FLAG              ;NO INTERRUPT
13962 070534  004767  000534                  JSR     R7,MLFSUB                   ;DO TEST
13963 070540  000200  000000                  .WORD   200,0   ;ACO
13964 070544  000000  000000                  .WORD   0,0     ;FSRC
13965 070550  000000  000000                  .WORD   0,0                 ;RESULT
13966 070554  000013                          .WORD   13                  ; TEST FPS
13967 070556  000004                          .WORD   4                   ;RESULTANT FPS
13968                                   ;3/AC=0 FSRC>0 -
13969 070560  005037  003030                  CLR     @#FLAG              ;NO INTERRUPT
13970 070564  004767  000504                  JSR     R7,MLFSUB                   ;DO TEST
13971 070570  000100  000000                  .WORD   100,0   ;ACO
13972 070574  000300  000000                  .WORD   300,0   ;FSRC
13973 070600  000000  000000                  .WORD   0,0                 ;RESULT
13974 070604  007500                          .WORD   7500                ; TEST FPS
13975 070606  007504                          .WORD   7504                ;RESULTANT FPS
13976                                   ;4/AC=1 >FSRC - ROUND
13977 070610  005037  003030                  CLR     @#FLAG              ;NO INTERRUPT
13978 070614  004767  000454                  JSR     R7,MLFSUB                   ;DO TEST
13979 070620  040200  000000                  .WORD   40200,0 ;ACO
13980 070624  040177  177777                  .WORD   40177,-1                ;FSRC
13981 070630  040177  177777                  .WORD   40177,-1                    ;RESULT
13982 070634  000000                          .WORD   0                   ; TEST FPS
13983 070636  000000                          .WORD   0                   ;RESULTANT FPS
13984                                   ;5/TRUNCATE
13985 070640  005037  003030                  CLR     @#FLAG              ;NO INTERRUPT
13986 070644  004767  000424                  JSR     R7,MLFSUB                   ;DO TEST
13987 070650  040177  177777                  .WORD   40177,-1                ;ACO
13988 070654  040200  000000                  .WORD   40200,0 ;FSRC
13989 070660  040177  177777                  .WORD   40177,-1                    ;RESULT
13990 070664  000040                          .WORD   40                  ; TEST FPS
13991 070666  000040                          .WORD   40                  ;RESULTANT FPS
13992                                   ;6/NORMALIZE
```

FLOATING POINT TESTS

```
 13993 070670  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
 13994 070674  004767  000374                JSR     R7,MLFSUB               ;DO TEST
 13995 070700  040100  000000                .WORD   40100,0 ;AC0
 13996 070704  040100  000000                .WORD   40100,0 ;FSRC
 13997 070710  040020  000000                .WORD   40020,0         ;RESULT
 13998 070714  000012                        .WORD   12              ; TEST FPS
 13999 070716  000000                        .WORD   0               ;RESULTANT FPS
 14000                              ;7/ROUND
 14001 070720  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
 14002 070724  004767  000344                JSR     R7,MLFSUB               ;DO TEST
 14003 070730  017500  000000                .WORD   17500,0 ;AC0
 14004 070734  023652  125252                .WORD   23652,125252    ;FSRC
 14005 070740  003177  177777                .WORD   3177,-1         ;RESULT
 14006 070744  007417                        .WORD   7417            ; TEST FPS
 14007 070746  007400                        .WORD   7400            ;RESULTANT FPS
 14008                              ;8/AC>0>FSRC ROUND
 14009 070750  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
 14010 070754  004767  000314                JSR     R7,MLFSUB               ;DO TEST
 14011 070760  040342  177777                .WORD   40342,-1        ;AC0
 14012 070764  176543  025252                .WORD   176543,025252   ;FSRC
 14013 070770  176711  067324                .WORD   176711,67324            ;RESULT
 14014 070774  007500                        .WORD   7500            ; TEST FPS
 14015 070776  007510                        .WORD   7510            ;RESULTANT FPS
 14016                              ;9/IAC<FSRC<0, ROUND
 14017 071000  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
 14018 071004  004767  000264                JSR     R7,MLFSUB               ;DO TEST
 14019 071010  144600  000000                .WORD   144600,0                ;AC0
 14020 071014  154000  000000                .WORD   154000,0                ;FSRC
 14021 071020  060400  000000                .WORD   60400,0         ;RESULT
 14022 071024  000017                        .WORD   17              ; TEST FPS
 14023 071026  000000                        .WORD   0               ;RESULT FPS
 14024                              ;10/AC<FSRC, ROUND
 14025 071030  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
 14026 071034  004767  000234                JSR     R7,MLFSUB               ;DO TEST
 14027 071040  060000  000000                .WORD   60000,0         ;AC0
 14028 071044  140377  177776                .WORD   140377,177776           ;FSRC
 14029 071050  160177  177776                .WORD   160177,177776           ;RESULT
 14030 071054  000017                        .WORD   17              ; TEST FPS
 14031 071056  000010                        .WORD   10              ;RESULT FPS
 14032                              ;11/AC>0>FSRC, TRUNCATE
 14033 071060  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
 14034 071064  004767  000204                JSR     R7,MLFSUB               ;DO TEST
 14035 071070  060000  000000                .WORD   60000,0         ;AC0
 14036 071074  140377  177776                .WORD   140377,177776           ;FSRC
 14037 071100  160177  177776                .WORD   160177,177776           ;RESULT
 14038 071104  007547                        .WORD   7547            ; TEST FPS
 14039 071106  007550                        .WORD   7550            ;RESULT FPS
 14040                              ;12/UNDERFLOW, NO INTERRUPTS
 14041 071110  012737  000002  003030        MOV     #2,@#FLAG               ;NO INTERRUPT
 14042 071116  004767  000152                JSR     R7,MLFSUB               ;DO TEST
 14043 071122  000200  000001                .WORD   200,1           ;AC0
 14044 071126  000200  000001                .WORD   200,1           ;FSRC
 14045 071132  040200  000002                .WORD   40200,2         ;RESULT
 14046 071136  042117                        .WORD   42117           ; TEST FPS
 14047 071140  142100                        .WORD   142100          ;RESULT FPS
 14048 071142  000012                        .WORD   12              ;FEC
 14049                              ;13/OVERFLOW, TRAP
```

FLOATING POINT TESTS

```
14050 071144  012737  000001  003030          MOV     #1,@#FLAG               ;INTERRUPT
14051 071152  004767  000116                  JSR     R7,MLFSUB               ;DO TEST
14052 071156  177777  177777                  .WORD   177777,-1               ;ACO
14053 071162  040300  000000                  .WORD   40300,0         ;FSRC
14054 071166  100077  177777                  .WORD   100077,-1               ;RESULT
14055 071172  001117                          .WORD   1117            ; TEST FPS
14056 071174  101116                          .WORD   101116          ;RESULT FPS
14057 071176  000010                          .WORD   10              ;FEC
14058                       ;14/OVERFLOW NO TRAP
14059 071200  012737  000002  003030          MOV     #2,@#FLAG               ;NO INTERRUPT
14060 071206  004767  000062                  JSR     R7,MLFSUB               ;DO TEST
14061 071212  077700  000000                  .WORD   77700,0         ;ACO
14062 071216  077700  000000                  .WORD   77700,0         ;FSRC
14063 071222  000000  000000                  .WORD   0,0             ;RESULT
14064 071226  040117                          .WORD   40117           ; TEST FPS
14065 071230  040106                          .WORD   40106           ;RESULT FPS
14066 071232  000010                          .WORD   10              ;FEC
14067                       ;15/UNDEFINED VARIABLE IN FSRC, TRAP ENABLED
14068 071234  012737  000001  003030          MOV     #1,@#FLAG               ;INTERRUPT
14069 071242  004767  000026                  JSR     R7,MLFSUB               ;DO TEST
14070 071246  123465  000000                  .WORD   123465,0                ;ACO
14071 071252  100022  000000                  .WORD   100022,0                ;FSRC
14072 071256  123465  000000                  .WORD   123465,0                ;RESULT
14073 071262  004000                          .WORD   4000            ; TEST FPS
14074 071264  104000                          .WORD   104000          ;RESULT FPS
14075 071266  000014                          .WORD   14              ;FEC
14076                       ;
14077 071270  000167  000242                  JMP     HOP12
14078                       ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14079                       ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14080                       ;
14081                       ;               ACO
14082                       ;               FSRC
14083                       ;               FPS BEFORE EXECUTION
14084                       ;               FPS AFTER EXECUTION
14085                       ;               (FEC)
14086                       ;
14087                       ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14088                       ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14089                       ;
14090 071274  012605                  MLFSUB: MOV     (SP)+,R5                ; RETURN ADDRESS TO USE AS POINTER
14091 071276  012737  071356  000244          MOV     #50$,@#FPVEC            ;REDIRECT TRAP VECTOR
14092 071304  012702  000200                  MOV     #200,R2                 ;SET TO DOUBLE MODE FOR LOAD
14093 071310  170102                          LDFPS   R2                      ;LOAD FPS
14094 071312  172415                          LDD     (R5),AC0                ;LOAD AC0 WITH TEST DATA
14095 071314  010504                          MOV     R5,R4                   ;POINT TO FSRC DATA
14096 071316  062704  000004                  ADD     #4,R4
14097 071322  016502  000014                  MOV     14(R5),R2               ;GET TEST FPS
14098 071326  170102                          LDFPS   R2                      ;LOAD TEST FPS
14099                       ;
14100 071330  171014                          MULF    (R4),AC0                ;*TEST INSTRUCTION
14101 071332  170001                  1$:     SETF                            ;WAIT FOR POSSIBLE FPA TRAP.
14102                       ;
14103                       ;INSTRUCTION DIDNT TRAP
14104                       ;
14105 071334  032737  000001  003030          BIT     #1,@#FLAG                       ;VERIFY A NO TRAP CONDITION
14106 071342  001426                          BEQ     2$                      ;BRANCH IF GOOD
```

FLOATING POINT TESTS

```
14107 071344  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14108 071350  104003                        ERROR   +3              ;FPP ERROR
14109                                                               ;INSTRUCTION SHOULD HAVE TRAPPED
14110 071352  000167  000042                JMP     2$                      ;REJOIN CODE
14111                                 ;
14112                                 ;INSTRUCTION TRAPPED
14113                                 ;
14114 071356  032737  000001  003030  50$:   BIT     #1,@#FLAG               ;SEE IF EXPECTING A TRAP
14115 071364  001005                        BNE     51$                     ;BRANCH IF EXPECTING A TRAP
14116 071366  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14117 071372  104003                        ERROR   +3              ;FPP ERROR
14118                                                               ;INSTRUCTION WASNT SUPPOSE TO TRAP
14119 071374  000167  000020                JMP     2$                      ;REJOIN CODE
14120 071400  012604                51$:    MOV     (SP)+,R4                ;SEE IF PC = INSTRUCTION
14121 071402  005726                        TST     (SP)+                   ;CLEAN UP STACK
14122 071404  022704  071332                CMP     #1$,R4                  ;
14123 071410  001403                        BEQ     2$                      ;BRANCH IF GOOD COMPARE
14124 071412  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14125 071416  104003                        ERROR   +3              ;FPP ERROR
14126                                                               ;PC WAS INCORRECT
14127                                 ;
14128                                 ;COMMON CODE FOR TRAP AND NO TRAP
14129                                 ;
14130 071420  170203                2$:     STFPS   R3                      ;SAVE FPS
14131 071422  012702  000200                MOV     #200,R2                 ;SET FPP TO DOUBLE
14132 071426  170102                        LDFPS   R2
14133 071430  012701  003142                MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
14134 071434  174011                        STD     AC0,(R1)                ;SAVE AC0  RESULT
14135 071436  026503  000016                CMP     16(R5),R3               ;VERIFY STATUS
14136 071442  001403                        BEQ     3$                      ;BRANCH IF GOOD
14137 071444  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14138 071450  104003                        ERROR   +3              ;FPP ERROR
14139                                                               ;BAD FPS
14140 071452  010504                3$:     MOV     R5,R4                   ;POINT TO EXPECTED DATA
14141 071454  062704  000010                ADD     #10,R4
14142 071460  004767  046376          4$:   JSR     R7,DATVFR                       ;VERIFY DATA
14143 071464  005767  111430                TST     COUNT
14144 071470  001403                        BEQ     5$                      ;BRANCH IF GOOD
14145 071472  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14146 071476  104003                        ERROR   +3              ;FPP ERROR
14147                                                               ;BAD AC0
14148 071500  005737  003030          5$:   TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
14149 071504  001002                        BNE     6$                      ;BRANCH IF NEED TO CHECK
14150 071506  000165  000020                JMP     20(R5)                  ;RETURN FROM TEST
14151                                 ;
14152                                 ;VERIFY ERROR STATUS
14153                                 ;
14154 071512  170301                6$:     STST    R1                      ;SAVE FEC
14155 071514  016504  000020                MOV     20(R5),R4               ;GET FEC
14156 071520  020401                        CMP     R4,R1                   ;VERIFY FEC
14157 071522  001403                        BEQ     7$                      ;BRANCH IF GOOD
14158 071524  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14159 071530  104003                        ERROR   +3              ;FPP ERROR
14160                                                                       ;BAD FEC
14161 071532  000165  000022          7$:   JMP     22(R5)          ;RETURN FROM TEST
14162 071536                          HOP12:
14165                                 ;
```

FLOATING POINT TESTS

```
14166                                   ;---------------------------------
14167                                   ;TEST MULD
14168                                   ;
14169 071536                            MMULD:
14170                                   ;
14171                                   ;1/AC=0
14172 071536  005037  003030                      CLR     @#FLAG          ;NO INTERRUPT
14173 071542  004767  000554                      JSR     R7,MLDSUB               ;DO TEST
14174 071546  000100  000000  000000              .WORD   100,0,0,0       ;AC0
14175 071556  000411  177777  000000              .WORD   411,-1,0,1      ;FSRC
14176 071566  000000  000000  000000              .WORD   0,0,0,0         ;RESULT
14177 071576  000200                            .WORD   200             ; TEST FPS
14178 071600  000204                            .WORD   204             ;RESULTANT FPS
14179                                   ;2/FSRC=0
14180 071602  005037  003030                      CLR     @#FLAG          ;NO INTERRUPT
14181 071606  004767  000510                      JSR     R7,MLDSUB               ;DO TEST
14182 071612  077777  000000  000000              .WORD   77777,0,0,0     ;AC0
14183 071622  000000  000000  000000              .WORD   0,0,0,0 ;FSRC
14184 071632  000000  000000  000000              .WORD   0,0,0,0         ;RESULT
14185 071642  007700                            .WORD   7700            ; TEST FPS
14186 071644  007704                            .WORD   7704            ;RESULTANT FPS
14187                                   ;3/AC=1
14188 071646  005037  003030                      CLR     @#FLAG          ;NO INTERRUPT
14189 071652  004767  000444                      JSR     R7,MLDSUB               ;DO TEST
14190 071656  040200  000000  000000              .WORD   40200,0,0,0     ;AC0
14191 071666  000277  177777  177777              .WORD   277,-1,-1,-1    ;FSRC
14192 071676  000277  177777  177777              .WORD   277,-1,-1,-1            ;RESULT
14193 071706  007717                            .WORD   7717            ; TEST FPS
14194 071710  007700                            .WORD   7700            ;RESULTANT FPS
14195                                   ;4/AC>FSRC>0, TRUNCATE
14196 071712  005037  003030                      CLR     @#FLAG          ;NO INTERRUPT
14197 071716  004767  000400                      JSR     R7,MLDSUB               ;DO TEST
14198 071722  065500  000000  000000              .WORD   65500,0,0,1     ;AC0
14199 071732  037577  177777  177777              .WORD   37577,-1,-1,-2  ;FSRC
14200 071742  065077  177777  177777              .WORD   65077,-1,-1,-1          ;RESULT
14201 071752  007717                            .WORD   7717            ; TEST FPS
14202 071754  007700                            .WORD   7700            ;RESULTANT FPS
14203                                   ;5/AC<FSRC<0
14204 071756  005037  003030                      CLR     @#FLAG          ;NO INTERRUPT
14205 071762  004767  000334                      JSR     R7,MLDSUB               ;DO TEST
14206 071766  137577  177777  177777              .WORD   137577,-1,-1,-2 ;AC0
14207 071776  165400  000000  000000              .WORD   165400,0,0,1    ;FSRC
14208 072006  065000  000000  000000              .WORD   65000,0,0,0             ;RESULT
14209 072016  007717                            .WORD   7717            ; TEST FPS
14210 072020  007700                            .WORD   7700            ;RESULTANT FPS
14211                                   ;6/AC>FSRC>0
14212 072022  005037  003030                      CLR     @#FLAG          ;NO INTERRUPT
14213 072026  004767  000270                      JSR     R7,MLDSUB               ;DO TEST
14214 072032  017500  000000  000000              .WORD   17500,0,0,0     ;AC0
14215 072042  123652  125252  125252              .WORD   123652,125252,125252,125252     ;FSRC
14216 072052  103177  177777  177777              .WORD   103177,-1,-1,-1         ;RESULT
14217 072062  000200                            .WORD   200             ; TEST FPS
14218 072064  000210                            .WORD   210             ;RESULTANT FPS
14219                                   ;7/UNDERFLOW, TRAPS DISABLED
14220 072066  005037  003030                      CLR     @#FLAG                  ;NO INTERRUPT
14221 072072  004767  000224                      JSR     R7,MLDSUB               ;DO TEST
14222 072076  000300  000000  000000              .WORD   300,0,0,252             ;AC0
```

FLOATING POINT TESTS

```
14223 072106  000377  000001  000002          .WORD   377,1,2,3                ;FSRC
14224 072116  000000  000000  000000          .WORD   0,0,0,0          ;RESULT
14225 072126  005740                          .WORD   5740             ; TEST FPS
14226 072130  005744                          .WORD   5744             ;RESULT FPS
14227                               ;8/UNDERFLOW, TRAP ENABLED
14228 072132  012737  000001  003030          MOV     #1,@#FLAG                ;INTERRUPT
14229 072140  004767  000156                  JSR     R7,MLDSUB                ;DO TEST
14230 072144  100277  000001  000002          .WORD   100277,1,2,-1            ;AC0
14231 072154  100300  000001  000001          .WORD   100300,1,1,1             ;FSRC
14232 072164  040417  040001  077403          .WORD   40417,40001,77403,0              ;RESULT
14233 072174  002217                          .WORD   2217             ; TEST FPS
14234 072176  102200                          .WORD   102200           ;RESULT FPS
14235 072200  000012                          .WORD   12               ;FEC
14236                               ;9/OVERFLOW, TRAPS DISABLED
14237 072202  005037  003030                  CLR     @#FLAG                   ;NO INTERRUPT
14238 072206  004767  000110                  JSR     R7,MLDSUB                ;DO TEST
14239 072212  177777  177777  177777          .WORD   -1,-1,-1,-1              ;AC0
14240 072222  040200  177777  177777          .WORD   40200,-1,-1,-1           ;FSRC
14241 072232  000000  000000  000000          .WORD   0,0,0,0          ;RESULT
14242 072242  006740                          .WORD   6740             ; TEST FPS
14243 072244  006746                          .WORD   6746             ;RESULT FPS
14244                               ;10/OVERFLOW, TRAPS ENABLED
14245 072246  012737  000001  003030          MOV     #1,@#FLAG                ;INTERRUPT
14246 072254  004767  000042                  JSR     R7,MLDSUB                ;DO TEST
14247 072260  157700  025252  025252          .WORD   157700,25252,25252,25252                 ;AC0
14248 072270  167700  000000  000000          .WORD   167700,0,0,0             ;FSRC
14249 072300  007420  017777  117777          .WORD   7420,017777,117777,117777                ;RESULT
14250 072310  001240                          .WORD   1240             ; TEST FPS
14251 072312  101242                          .WORD   101242           ;RESULT FPS
14252 072314  000010                          .WORD   10               ;FEC
14253                               ;
14254 072316  000167  000242                  JMP     HOP13
14255                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14256                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14257                               ;
14258                               ;               AC0
14259                               ;               FSRC
14260                               ;               FPS BEFORE EXECUTION
14261                               ;               FPS AFTER EXECUTION
14262                               ;               (FEC)
14263                               ;
14264                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14265                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14266                               ;
14267 072322  012605                MLDSUB: MOV     (SP)+,R5                 ; RETURN ADDRESS TO USE AS POINTER
14268 072324  012737  072404  000244          MOV     #50$,@#FPVEC             ;REDIRECT TRAP VECTOR
14269 072332  012702  000200                  MOV     #200,R2                  ;SET TO DOUBLE MODE FOR LOAD
14270 072336  170102                          LDFPS   R2                       ;LOAD FPS
14271 072340  172415                          LDD     (R5),AC0                 ;LOAD AC0 WITH TEST DATA
14272 072342  010501                          MOV     R5,R1                    ;POINT TO FSRC DATA
14273 072344  062701  000010                  ADD     #10,R1
14274 072350  016502  000030                  MOV     30(R5),R2                ;GET TEST FPS
14275 072354  170102                          LDFPS   R2                       ;LOAD TEST FPS
14276                               ;
14277 072356  171011                          MULD    (R1),AC0                 ;*TEST INSTRUCTION
14278 072360  170011                1$:       SETD                             ;WAIT FOR POSSIBLE FPA TRAP.
14279                               ;
```

FLOATING POINT TESTS

```
14280                                   ;INSTRUCTION DIDNT TRAP
14281                                   ;
14282 072362  032737  000001  003030          BIT     #1,@#FLAG                       ;VERIFY A NO TRAP CONDITION
14283 072370  001426                          BEQ     2$                      ;BRANCH IF GOOD
14284 072372  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14285 072376  104003                          ERROR   +3                      ;FPP ERROR
14286                                                                         ;INSTRUCTION SHOULD HAVE TRAPPED
14287 072400  000167  000042                  JMP     2$                      ;REJOIN CODE
14288                                   ;
14289                                   ;INSTRUCTION TRAPPED
14290                                   ;
14291 072404  032737  000001  003030  50$:    BIT     #1,@#FLAG               ;SEE IF EXPECTING A TRAP
14292 072412  001005                          BNE     51$                     ;BRANCH IF EXPECTING A TRAP
14293 072414  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14294 072420  104003                          ERROR   +3                      ;FPP ERROR
14295                                                                         ;INSTRUCTION WASNT SUPPOSE TO TRAP
14296 072422  000167  000020                  JMP     2$                      ;REJOIN CODE
14297 072426  012604                  51$:    MOV     (SP)+,R4                ;SEE IF PC = INSTRUCTION
14298 072430  005726                          TST     (SP)+                   ;CLEAN UP STACK
14299 072432  022704  072360                  CMP     #1$,R4                  ;
14300 072436  001403                          BEQ     2$                      ;BRANCH IF GOOD COMPARE
14301 072440  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14302 072444  104003                          ERROR   +3                      ;FPP ERROR
14303                                                                         ;PC WAS INCORRECT
14304                                   ;
14305                                   ;COMMON CODE FOR TRAP AND NO TRAP
14306                                   ;
14307 072446  170203                  2$:     STFPS   R3                      ;SAVE FPS
14308 072450  012702  000200                  MOV     #200,R2                 ;SET FPP TO DOUBLE
14309 072454  170102                          LDFPS   R2
14310 072456  012701  003142                  MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
14311 072462  174011                          STD     AC0,(R1)                ;SAVE AC0  RESULT
14312 072464  026503  000032                  CMP     32(R5),R3               ;VERIFY STATUS
14313 072470  001403                          BEQ     3$                      ;BRANCH IF GOOD
14314 072472  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14315 072476  104003                          ERROR   +3                      ;FPP ERROR
14316                                                                         ;BAD FPS
14317 072500  010504                  3$:     MOV     R5,R4                   ;POINT TO EXPECTED DATA
14318 072502  062704  000020                  ADD     #20,R4
14319 072506  004767  045366          4$:     JSR     R7,DATVER                       ;VERIFY DATA
14320 072512  005767  110402                  TST     COUNT
14321 072516  001403                          BEQ     5$                      ;BRANCH IF GOOD
14322 072520  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14323 072524  104003                          ERROR   +3                      ;FPP ERROR
14324                                                                         ;BAD AC0
14325 072526  005737  003030          5$:     TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
14326 072532  001002                          BNE     6$                      ;BRANCH IF NEED TO CHECK
14327 072534  000165  000034                  JMP     34(R5)                  ;RETURN FROM TEST
14328                                   ;VERIFY ERROR STATUS
14329 072540  170301                  6$:     STST    R1                      ;SAVE FEC
14330 072542  016504  000034                  MOV     34(R5),R4               ;GET FEC
14331 072546  020401                          CMP     R4,R1                   ;VERIFY FEC
14332 072550  001403                          BEQ     7$                      ;BRANCH IF GOOD
14333 072552  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14334 072556  104003                          ERROR   +3                      ;FPP ERROR
14335                                                                         ;BAD FEC
14336 072560  000165  000036          7$:     JMP     36(R5)                  ;RETURN FROM TEST
```

```
14337 072564                          HOP13:
14340                                 ;
14341                                 ;---------------------------------
14342                                 ;TEST MODF
14343                                 ;
14344 072564                          MMODF:
14345                                 ;
14346                                 ;1/AC=0 FSRC=0
14347 072564  005037  003030                  CLR     @#FLAG          ;NO INTERRUPT
14348 072570  004767  000554                  JSR     R7,MDFSUB               ;DO TEST
14349 072574  000100  000000                  .WORD   100,0   ;AC0
14350 072600  012346  177777                  .WORD   12346,-1        ;FSRC
14351 072604  000000  000000                  .WORD   0,0             ;FRACTIONAL RESULT
14352 072610  000000  000000                  .WORD   0,0             ;INTEGER RESULT
14353 072614  000013                          .WORD   13              ; TEST FPS
14354 072616  000004                          .WORD   4               ;RESULTANT FPS
14355                                 ;2/FSRC=0
14356 072620  005037  003030                  CLR     @#FLAG          ;NO INTERRUPT
14357 072624  004767  000520                  JSR     R7,MDFSUB               ;DO TEST
14358 072630  012356  177777                  .WORD   12356,-1        ;AC0
14359 072634  000000  000000                  .WORD   0,0     ;FSRC
14360 072640  000000  000000                  .WORD   0,0             ;FRACTIONAL RESULT
14361 072644  000000  000000                  .WORD   0,0             ;INTEGER RESULT
14362 072650  000003                          .WORD   3               ; TEST FPS
14363 072652  000004                          .WORD   4       ;RESULTANT FPS
14364                                 ;3/AC=0
14365 072654  005037  003030                  CLR     @#FLAG          ;NO INTERRUPT
14366 072660  004767  000464                  JSR     R7,MDFSUB               ;DO TEST
14367 072664  000000  000000                  .WORD   0,0             ;AC0
14368 072670  177777  177777                  .WORD   -1,-1           ;FSRC
14369 072674  000000  000000                  .WORD   0,0             ;FRACTIONAL  RESULT
14370 072700  000000  000000                  .WORD   0,0             ;INTEGER RESULT
14371 072704  007500                          .WORD   7500            ; TEST FPS
14372 072706  007504                          .WORD   7504            ;RESULT FPS
14373                                 ;4/AC>FSRC>0
14374 072710  005037  003030                  CLR     @#FLAG          ;NO INTERRUPT
14375 072714  004767  000430                  JSR     R7,MDFSUB               ;DO TEST
14376 072720  046252  125252                  .WORD   46252,125252    ;AC0
14377 072724  040300  000000                  .WORD   40300,0 ;FSRC
14378 072730  000000  000000                  .WORD   0,0             ;FRACTIONAL RESULT
14379 072734  046377  177777                  .WORD   46377,-1                ;INTEGER RESULT
14380 072740  000013                          .WORD   13              ; TEST FPS
14381 072742  000004                          .WORD   4               ;RESULTANT FPS
14382                                 ;5/AC>FSRC>0
14383 072744  005037  003030                  CLR     @#FLAG          ;NO INTERRUPT
14384 072750  004767  000374                  JSR     R7,MDFSUB               ;DO TEST
14385 072754  077652  125252                  .WORD   77652,125252    ;AC0
14386 072760  040300  000000                  .WORD   40300,0 ;FSRC
14387 072764  000000  000000                  .WORD   0,0             ;FRACTIONAL RESULT
14388 072770  077777  177777                  .WORD   77777,-1                ;INTEGER RESULT
14389 072774  000000                          .WORD   0               ; TEST FPS
14390 072776  000004                          .WORD   4               ;RESULTANT FPS
14391                                 ;6/AC>0<FSRC, INTEGERS
14392 073000  005037  003030                  CLR     @#FLAG          ;NO INTERRUPT
14393 073004  004767  000340                  JSR     R7,MDFSUB               ;DO TEST
14394 073010  060600  000000                  .WORD   60600,0         ;AC0
14395 073014  147400  025700                  .WORD   147400,25700            ;FSRC
```

FLOATING POINT TESTS

```
14396 073020  000000  000000                .WORD   0,0                 ;FRACTIONAL  RESULT
14397 073024  170000  025700                .WORD   170000,25700                ;INTEGER RESULT
14398 073030  007400                        .WORD   7400                ; TEST FPS
14399 073032  007404                        .WORD   7404                ;RESULT FPS
14400                             ;7/AC<0<FSRC, FRACTIONAL
14401 073034  005037  003030                CLR     @#FLAG              ;NO INTERRUPT
14402 073040  004767  000304                JSR     R7,MDFSUB                   ;DO TEST
14403 073044  100227  177777                .WORD   100227,-1                   ;ACO
14404 073050  044025  025252                .WORD   44025,25252                 ;FSRC
14405 073054  104061  021251                .WORD   104061,21251                ;FRACTIONAL  RESULT
14406 073060  000000  000000                .WORD   0,0                 ;INTEGER RESULT
14407 073064  000000                        .WORD   0                   ; TEST FPS
14408 073066  000010                        .WORD   10                  ;RESULT FPS
14409                             ;8/AC<0>FSRC, TRUNCATE
14410 073070  005037  003030                CLR     @#FLAG              ;NO INTERRUPT
14411 073074  004767  000250                JSR     R7,MDFSUB                   ;DO TEST
14412 073100  046252  125252                .WORD   46252,125252                ;ACO
14413 073104  040300  000000                .WORD   40300,0             ;FSRC
14414 073110  000000  000000                .WORD   0,0                 ;FRACTIONAL  RESULT
14415 073114  046377  177777                .WORD   46377,-1                    ;INTEGER RESULT
14416 073120  000053                        .WORD   53                  ; TEST FPS
14417 073122  000044                        .WORD   44                  ;RESULT FPS
14418                             ;9/ROUND INTEGER
14419 073124  005037  003030                CLR     @#FLAG              ;NO INTERRUPT
14420 073130  004767  000214                JSR     R7,MDFSUB                   ;DO TEST
14421 073134  046252  125252                .WORD   46252,125252                ;ACO
14422 073140  040300  000000                .WORD   40300,0             ;FSRC
14423 073144  000000  000000                .WORD   0,0                 ;FRACTIONAL  RESULT
14424 073150  046377  177777                .WORD   46377,-1                    ;INTEGER RESULT
14425 073154  000013                        .WORD   13                  ; TEST FPS
14426 073156  000004                        .WORD   4                   ;RESULT FPS
14427                             ;10/TRUNCATE FRACTION
14428 073160  005037  003030                CLR     @#FLAG              ;NO INTERRUPT
14429 073164  004767  000160                JSR     R7,MDFSUB                   ;DO TEST
14430 073170  040777  177777                .WORD   40777,-1                    ;ACO
14431 073174  040200  000000                .WORD   40200,0             ;FSRC
14432 073200  040177  177770                .WORD   40177,177770                ;FRACTIONAL  RESULT
14433 073204  040740  000000                .WORD   40740,0             ;INTEGER RESULT
14434 073210  000000                        .WORD   0                   ; TEST FPS
14435 073212  000000                        .WORD   0                   ;RESULT FPS
14436                             ;11/ROUND INTEGER
14437 073214  005037  003030                CLR     @#FLAG              ;NO INTERRUPT
14438 073220  004767  000124                JSR     R7,MDFSUB                   ;DO TEST
14439 073224  000000  000000                .WORD   0,0                 ;ACO
14440 073230  000000  000000                .WORD   0,0                 ;FSRC
14441 073234  000000  000000                .WORD   0,0                 ;FRACTIONAL  RESULT
14442 073240  000000  000000                .WORD   0,0                 ;INTEGER RESULT
14443 073244  000000                        .WORD   0                   ; TEST FPS
14444 073246  000004                        .WORD   4                   ;RESULT FPS
14445                             ;12/ROUND FRACTION
14446 073250  005037  003030                CLR     @#FLAG              ;NO INTERRUPT
14447 073254  004767  000070                JSR     R7,MDFSUB                   ;DO TEST
14448 073260  040225  125252                .WORD   40225,125252                ;ACO
14449 073264  066652  052525                .WORD   66652,52525                 ;FSRC
14450 073270  000000  000000                .WORD   0,0                 ;FRACTIONAL  RESULT
14451 073274  066707  025160                .WORD   66707,25160                 ;INTEGER RESULT
14452 073300  007027                        .WORD   7027                ; TEST FPS
```

FLOATING POINT TESTS

```
14453 073302  007004                          .WORD   7004            ;RESULT FPS
14454                               ;/OVERFLOW
14455 073304  012737  000001  003030          MOV     #1,@#FLAG               ;INTERRUPT
14456 073312  004767  000032                  JSR     R7,MDFSUB               ;DO TEST
14457 073316  076000  000000                  .WORD   76000,0         ;ACO
14458 073322  076000  000000                  .WORD   76000,0         ;FSRC
14459 073326  000000  000000                  .WORD   0,0             ;FRACTIONAL  RESULT
14460 073332  033600  000000                  .WORD   33600,0         ;INTEGER RESULT
14461 073336  001000                          .WORD   1000            ; TEST FPS
14462 073340  101006                          .WORD   101006          ;RESULT FPS
14463 073342  000010                          .WORD   10              ;FEC
14464                               ;
14465 073344  000167  000310                  JMP     HOP14
14466                               ;
14467                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14468                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14469                               ;
14470                               ;               ACO
14471                               ;               FSRC
14472                               ;               FRACTIONAL RESULT
14473                               ;               INTEGER RESULT
14474                               ;               FPS BEFORE EXECUTION
14475                               ;               FPS AFTER EXECUTION
14476                               ;               (FEC)
14477                               ;
14478                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14479                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14480                               ;
14481 073350  012605                  MDFSUB: MOV     (SP)+,R5                ; RETURN ADDRESS TO USE AS POINTER
14482 073352  012737  073440  000244          MOV     #50$,@#FPVEC            ;REDIRECT TRAP VECTOR
14483 073360  012702  000200                  MOV     #200,R2                 ;SET TO DOUBLE MODE FOR LOAD
14484 073364  170102                          LDFPS   R2                      ;LOAD FPS
14485 073366  172415                          LDD     (R5),AC0                ;LOAD AC0 WITH TEST DATA
14486 073370  012701  073650                  MOV     #MODGAR,R1              ;LOAD KNOWN INTO AC1
14487 073374  172511                          LDD     (R1),AC1                ;
14488 073376  010501                          MOV     R5,R1                   ;POINT TO FSRC DATA
14489 073400  062701  000004                  ADD     #4,R1
14490 073404  016502  000020                  MOV     20(R5),R2               ;GET TEST FPS
14491 073410  170102                          LDFPS   R2                      ;LOAD TEST FPS
14492                               ;
14493 073412  171411                          MODF    (R1),AC0                ;*TEST INSTRUCTION
14494 073414  170001                  1$:     SETF                            ;WAIT FOR POSSIBLE FPA TRAP.
14495                               ;
14496                               ;INSTRUCTION DIDNT TRAP
14497                               ;
14498 073416  032737  000001  003030          BIT     #1,@#FLAG                       ;VERIFY A NO TRAP CONDITION
14499 073424  001426                          BEQ     2$                      ;BRANCH IF GOOD
14500 073426  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14501 073432  104003                          ERROR   +3              ;FPP ERROR
14502                                                                 ;INSTRUCTION SHOULD HAVE TRAPPED
14503 073434  000167  000042                  JMP     2$                      ;REJOIN CODE
14504                               ;
14505                               ;INSTRUCTION TRAPPED
14506                               ;
14507 073440  032737  000001  003030  50$:     BIT     #1,@#FLAG               ;SEE IF EXPECTING A TRAP
14508 073446  001005                          BNE     51$                     ;BRANCH IF EXPECTING A TRAP
14509 073450  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
```

FLOATING POINT TESTS

```
14510 073454  104003                    ERROR   +3              ;FPP ERROR
14511                                                           ;INSTRUCTION WASNT SUPPOSE TO TRAP
14512 073456  000167  000020             JMP     2$                      ;REJOIN CODE
14513 073462  012604              51$:   MOV     (SP)+,R4                ;SEE IF PC = INSTRUCTION
14514 073464  005726                     TST     (SP)+                   ;CLEAN UP STACK
14515 073466  022704  073414             CMP     #1$,R4                  ;
14516 073472  001403                     BEQ     2$                      ;BRANCH IF GOOD COMPARE
14517 073474  004737  140132             CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14518 073500  104003                     ERROR   +3              ;FPP ERROR
14519                                                           ;PC WAS INCORRECT
14520                                   ;
14521                                   ;COMMON CODE FOR TRAP AND NO TRAP
14522                                   ;
14523 073502  170203              2$:    STFPS   R3                      ;SAVE FPS
14524 073504  012702  000200             MOV     #200,R2                 ;SET FPP TO DOUBLE
14525 073510  170102                     LDFPS   R2
14526 073512  012701  003142             MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
14527                                   ;SAVE FRACTIONAL RESULT
14528 073516  174011                     STD     AC0,(R1)                ;SAVE AC0  RESULT
14529 073520  026503  000022             CMP     22(R5),R3               ;VERIFY STATUS
14530 073524  001403                     BEQ     3$                      ;BRANCH IF GOOD
14531 073526  004737  140132             CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14532 073532  104003                     ERROR   +3              ;FPP ERROR
14533                                                           ;BAD FPS
14534 073534  010504              3$:    MOV     R5,R4                   ;POINT TO EXPECTED DATA
14535 073536  062704  000010             ADD     #10,R4
14536 073542  004767  044314      4$:    JSR     R7,DATVFR                       ;VERIFY DATA
14537 073546  005767  107346             TST     COUNT
14538 073552  001403                     BEQ     5$                      ;BRANCH IF GOOD
14539 073554  004737  140132             CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14540 073560  104003                     ERROR   +3              ;FPP ERROR
14541                                                           ;BAD AC0
14542                                   ;SAVE INTEGER RESULT
14543 073562  174111              5$:    STD     AC1,(R1)                ;SAVE AC1 RESULT
14544 073564  010504                     MOV     R5,R4           ;POINT TO EXPECTED
14545 073566  062704  000014             ADD     #14,R4
14546 073572  004767  044264             JSR     R7,DATVFR               ;VERIFY DATA
14547 073576  005767  107316             TST     COUNT                   ;
14548 073602  001403                     BEQ     6$                      ;BRANCH IF GOOD
14549 073604  004737  140132             CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14550 073610  104003                     ERROR   +3              ;FPP ERROR
14551                                                           ;BAD AC1
14552 073612  005737  003030      6$:    TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
14553 073616  001002                     BNE     7$                      ;BRANCH IF NEED TO CHECK
14554 073620  000165  000024             JMP     24(R5)                  ;RETURN FROM TEST
14555 073624  170301              7$:    STST    R1                      ;SAVE FEC
14556 073626  016504  000024             MOV     24(R5),R4               ;GET FEC
14557 073632  020401                     CMP     R4,R1                   ;VERIFY FEC
14558 073634  001403                     BEQ     8$                      ;BRANCH IF GOOD
14559 073636  004737  140132             CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14560 073642  104003                     ERROR   +3              ;FPP ERROR
14561                                                                   ;BAD FEC
14562 073644  000165  000026      8$:    JMP     26(R5)          ;RETURN FROM TEST
14563                                   ;
14564 073650  177777  177777  177777 MODGAR: .WORD   -1,-1,-1,-1             ;KNOWN DATA FOR AC1
14565 073660                           HOP14:
14566
```

```
14569                                   ;
14570                                   ;---------------------------------
14571                                   ;TEST MODD
14572                                   ;
14573 073660                            MMODD:
14574                                   ;
14575                                   ;1/AC>FSRC=0
14576 073660  005037  003030                      CLR     @#FLAG          ;NO INTERRUPT
14577 073664  004767  001164                      JSR     R7,MDDSUB               ;DO TEST
14578 073670  012345  177777  177777            .WORD   12345,-1,-1,-1  ;AC0
14579 073700  000100  000000  000000            .WORD   100,0,0,0       ;FSRC
14580 073710  000000  000000  000000            .WORD   0,0,0,0         ;FRACTIONAL RESULT
14581 073720  000000  000000  000000            .WORD   0,0,0,0         ;INTEGER RESULT
14582 073730  000200                            .WORD   200             ; TEST FPS
14583 073732  000204                            .WORD   204             ;RESULTANT FPS
14584                                   ;2/AC=0<FSRC
14585 073734  005037  003030                      CLR     @#FLAG          ;NO INTERRUPT
14586 073740  004767  001110                      JSR     R7,MDDSUB               ;DO TEST
14587 073744  000000  000000  000000            .WORD   0,0,0,0 ;AC0
14588 073754  001234  177777  000000            .WORD   1234,-1,0,0     ;FSRC
14589 073764  000000  000000  000000            .WORD   0,0,0,0         ;FRACTIONAL RESULT
14590 073774  000000  000000  000000            .WORD   0,0,0,0         ;INTEGER RESULT
14591 074004  007717                            .WORD   7717            ; TEST FPS
14592 074006  007704                            .WORD   7704            ;RESULTANT FPS
14593                                   ;3/AC>FSRC>0
14594 074010  005037  003030                      CLR     @#FLAG          ;NO INTERRUPT
14595 074014  004767  001034                      JSR     R7,MDDSUB               ;DO TEST
14596 074020  056252  125252  125252            .WORD   56252,125252,125252,125250      ;AC0
14597 074030  040300  000000  000000            .WORD   40300,0,0,0     ;FSRC
14598 074040  000000  000000  000000            .WORD   0,0,0,0         ;FRACTIONAL RESULT
14599 074050  056377  177777  177777            .WORD   56377,-1,-1,-4          ;INTEGER RESULT
14600 074060  000213                            .WORD   213             ; TEST FPS
14601 074062  000204                            .WORD   204             ;RESULTANT FPS
14602                                   ;4/AC<0>FSRC
14603 074064  005037  003030                      CLR     @#FLAG          ;NO INTERRUPT
14604 074070  004767  000760                      JSR     R7,MDDSUB               ;DO TEST
14605 074074  140240  000000  000000            .WORD   140240,0,0,0    ;AC0
14606 074104  063714  146314  133572            .WORD   63714,146314,133572,167737      ;FSRC
14607 074114  000000  000000  000000            .WORD   0,0,0,0         ;FRACTIONAL RESULT
14608 074124  163777  177777  162531            .WORD   163777,-1,162531,125726         ;INTEGER RESULT
14609 074134  000210                            .WORD   210             ; TEST FPS
14610 074136  000204                            .WORD   204             ;RESULTANT FPS
14611                                   ;5/AC>FSRC>0
14612 074140  005037  003030                      CLR     @#FLAG          ;NO INTERRUPT
14613 074144  004767  000704                      JSR     R7,MDDSUB               ;DO TEST
14614 074150  056200  000000  000000            .WORD   56200,0,0,1     ;AC0
14615 074160  040340  000000  000000            .WORD   40340,0,0,0     ;FSRC
14616 074170  000000  000000  000000            .WORD   0,0,0,0         ;FRACTIONAL RESULT
14617 074200  056340  000000  000000            .WORD   56340,0,0,1             ;INTEGER RESULT
14618 074210  000213                            .WORD   213             ; TEST FPS
14619 074212  000204                            .WORD   204             ;RESULTANT FPS
14620                                   ;6/TRUNCATE
14621 074214  005037  003030                      CLR     @#FLAG          ;NO INTERRUPT
14622 074220  004767  000630                      JSR     R7,MDDSUB               ;DO TEST
14623 074224  056252  125252  125252            .WORD   56252,125252,125252,125252              ;AC0
14624 074234  040300  000000  000000            .WORD   40300,0,0,0             ;FSRC
14625 074244  000000  000000  000000            .WORD   0,0,0,0         ;FRACTIONAL  RESULT
```

FLOATING POINT TESTS

```
14626 074254  056377  177777  177777          .WORD   56377,-1,-1,-1          ;INTEGER RESULT
14627 074264  000253                           .WORD   253                ; TEST FPS
14628 074266  000244                           .WORD   244                ;RESULT FPS
14629                               ;7/TRUNCATE FRACTION
14630 074270  005037  003030                   CLR     @#FLAG             ;NO INTERRUPT
14631 074274  004767  000554                   JSR     R7,MDDSUB                  ;DO TEST
14632 074300  023252  125252  125252          .WORD   23252,125252,125252,125252              ;ACO
14633 074310  040300  000000  000000          .WORD   40300,0,0,0                ;FSRC
14634 074320  023377  177777  177777          .WORD   23377,-1,-1,-1             ;FRACTIONAL  RESULT
14635 074330  000000  000000  000000          .WORD   0,0,0,0            ;INTEGER RESULT
14636 074340  000253                           .WORD   253                ; TEST FPS
14637 074342  000240                           .WORD   240                ;RESULT FPS
14638                               ;8/ROUND INTEGER
14639 074344  005037  003030                   CLR     @#FLAG             ;NO INTERRUPT
14640 074350  004767  000500                   JSR     R7,MDDSUB                  ;DO TEST
14641 074354  076600  000000  000000          .WORD   76600,0,0,125252                ;ACO
14642 074364  040300  000000  000000          .WORD   40300,0,0,0                ;FSRC
14643 074374  000000  000000  000000          .WORD   0,0,0,0            ;FRACTIONAL  RESULT
14644 074404  076700  000000  000000          .WORD   76700,0,0,-1               ;INTEGER RESULT
14645 074414  000200                           .WORD   200                ; TEST FPS
14646 074416  000204                           .WORD   204                ;RESULT FPS
14647                               ;9/ROUND THROUGH FRACTION
14648 074420  005037  003030                   CLR     @#FLAG             ;NO INTERRUPT
14649 074424  004767  000424                   JSR     R7,MDDSUB                  ;DO TEST
14650 074430  041525  052525  052525          .WORD   41525,052525,52525,52525                ;ACO
14651 074440  040300  000000  000000          .WORD   40300,0,0,0                ;FSRC
14652 074450  040177  177777  177777          .WORD   40177,-1,-1,177740              ;FRACTIONAL  RESULT
14653 074460  041636  000000  000000          .WORD   41636,0,0,0                ;INTEGER RESULT
14654 074470  007700                           .WORD   7700               ; TEST FPS
14655 074472  007700                           .WORD   7700               ;RESULT FPS
14656                               ;/OVERFLOW, TRAPS ENABLED
14657 074474  012737  000001  003030          MOV     #1,@#FLAG                  ;INTERRUPT
14658 074502  004767  000346                   JSR     R7,MDDSUB                  ;DO TEST
14659 074506  177777  177777  177777          .WORD   -1,-1,-1,-1                ;ACO
14660 074516  040400  000000  000000          .WORD   40400,0,0,0                ;FSRC
14661 074526  000000  000000  000000          .WORD   0,0,0,0            ;FRACTIONAL  RESULT
14662 074536  100177  177777  177777          .WORD   100177,-1,-1,-1            ;INTEGER RESULT
14663 074546  007700                           .WORD   7700               ; TEST FPS
14664 074550  107706                           .WORD   107706             ;RESULT FPS
14665 074552  000010                           .WORD   10                 ;FEC
14666                               ;/INTEGER CHOPPED TO 56 BITS
14667 074554  005037  003030                   CLR     @#FLAG             ;NO INTERRUPT
14668 074560  004767  000270                   JSR     R7,MDDSUB                  ;DO TEST
14669 074564  056700  000000  000000          .WORD   56700,0,0,-1               ;ACO
14670 074574  044440  177777  177777          .WORD   44440,-1,-1,-1             ;FSRC
14671 074604  000000  000000  000000          .WORD   0,0,0,0            ;FRACTIONAL  RESULT
14672 074614  063161  100000  000001          .WORD   63161,100000,1,40775            ;INTEGER RESULT
14673 074624  000200                           .WORD   200                ; TEST FPS
14674 074626  000204                           .WORD   204                ;RESULT FPS
14675                               ;/OVERFLOW, TRAPS DISABLED
14676 074630  012737  000002  003030          MOV     #2,@#FLAG                  ;NO INTERRUPT
14677 074636  004767  000212                   JSR     R7,MDDSUB                  ;DO TEST
14678 074642  066600  000000  000000          .WORD   66600,0,0,0                ;ACO
14679 074652  066600  000000  000000          .WORD   66600,0,0,0                ;FSRC
14680 074662  000000  000000  000000          .WORD   0,0,0,0            ;FRACTIONAL  RESULT
14681 074672  015200  000000  000000          .WORD   15200,0,0,0                ;INTEGER RESULT
14682 074702  047700                           .WORD   47700              ; TEST FPS
```

FLOATING POINT TESTS

```
14683 074704  147706                        .WORD   147706          ;RESULT FPS
14684 074706  000010                        .WORD   10              ;FEC
14685                               ;/UNDERFLOW, TRAPS DISABLED
14686 074710  012737  000002  003030        MOV     #2,@#FLAG               ;NO INTERRUPT
14687 074716  004767  000132                JSR     R7,MDDSUB               ;DO TEST
14688 074722  100277  000001  000002        .WORD   100277,1,2,-1           ;AC0
14689 074732  100300  000001  000001        .WORD   100300,1,1,1            ;FSRC
14690 074742  000000  000000  000000        .WORD   0,0,0,0         ;FRACTIONAL  RESULT
14691 074752  000000  000000  000000        .WORD   0,0,0,0         ;INTEGER RESULT
14692 074762  005200                        .WORD   5200            ; TEST FPS
14693 074764  005204                        .WORD   5204            ;RESULT FPS
14694 074766  000010                        .WORD   10              ;FEC
14695                               ;/UNDERFLOW TRAPS ENABLED, UV AS RESULT
14696 074770  012737  000001  003030        MOV     #1,@#FLAG               ;INTERRUPT
14697 074776  004767  000052                JSR     R7,MDDSUB               ;DO TEST
14698 075002  100277  000001  000002        .WORD   100277,1,2,-1           ;AC0
14699 075012  100300  000001  000001        .WORD   100300,1,1,1            ;FSRC
14700 075022  040417  040001  077403        .WORD   40417,40001,77403,0             ;FRACTIONAL  RESULT
14701 075032  000000  000000  000000        .WORD   0,0,0,0         ;INTEGER RESULT
14702 075042  002200                        .WORD   2200            ; TEST FPS
14703 075044  102200                        .WORD   102200          ;RESULT FPS
14704 075046  000012                        .WORD   12              ;FEC
14705                               ;
14706 075050  000167  000300                JMP     HOP15           ;JUMP OVER SUBROUTINE
14707                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14708                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14709                               ;
14710                               ;               AC0
14711                               ;               FSRC
14712                               ;               FRACTIONAL RESULT
14713                               ;               INTEGER RESULT
14714                               ;               FPS BEFORE EXECUTION
14715                               ;               FPS AFTER EXECUTION
14716                               ;               (FEC)
14717                               ;
14718                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14719                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14720                               ;
14721 075054  012605                MDDSUB: MOV     (SP)+,R5                ; RETURN ADDRESS TO USE AS POINTER
14722 075056  012737  075144  000244        MOV     #50$,@#FPVEC            ;REDIRECT TRAP VECTOR
14723 075064  012702  000200                MOV     #200,R2                 ;SET TO DOUBLE MODE FOR LOAD
14724 075070  170102                        LDFPS   R2                      ;LOAD FPS
14725 075072  172415                        LDD     (R5),AC0                ;LOAD AC0 WITH TEST DATA
14726 075074  012701  073650                MOV     #MODGAR,R1              ;LOAD KNOWN INTO AC1
14727 075100  172511                        LDD     (R1),AC1                ;
14728 075102  010501                        MOV     R5,R1                   ;POINT TO FSRC DATA
14729 075104  062701  000010                ADD     #10,R1
14730 075110  016502  000040                MOV     40(R5),R2               ;GET TEST FPS
14731 075114  170102                        LDFPS   R2                      ;LOAD TEST FPS
14732                               ;
14733 075116  171411                        MODD    (R1),AC0                ;*TEST INSTRUCTION
14734 075120  170011                1$:     SETD                            ;WAIT FOR POSSIBLE FPA TRAP.
14735                               ;
14736                               ;INSTRUCTION DIDNT TRAP
14737                               ;
14738 075122  032737  000001  003030        BIT     #1,@#FLAG                       ;VERIFY A NO TRAP CONDITION
14739 075130  001426                        BEQ     2$                      ;BRANCH IF GOOD
```

FLOATING POINT TESTS

```
14740 075132  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14741 075136  104003                        ERROR   .+3                     ;FPP ERROR
14742                                                                       ;INSTRUCTION SHOULD HAVE TRAPPED
14743 075140  000167  000042                JMP     2$                      ;REJOIN CODE
14744                                 ;
14745                                 ;INSTRUCTION TRAPPED
14746                                 ;
14747 075144  032737  000001  003030  50$:   BIT     #1,@#FLAG               ;SEE IF EXPECTING A TRAP
14748 075152  001005                        BNE     51$                     ;BRANCH IF EXPECTING A TRAP
14749 075154  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14750 075160  104003                        ERROR   .+3                     ;FPP ERROR
14751                                                                       ;INSTRUCTION WASNT SUPPOSE TO TRAP
14752 075162  000167  000020                JMP     2$                      ;REJOIN CODE
14753 075166  012604                  51$:   MOV     (SP)+,R4                ;SEE IF PC = INSTRUCTION
14754 075170  005726                        TST     (SP)+                   ;CLEAN UP STACK
14755 075172  022704  075120                CMP     #1$,R4                  ;
14756 075176  001403                        BEQ     2$                      ;BRANCH IF GOOD COMPARE
14757 075200  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14758 075204  104003                        ERROR   .+3                     ;FPP ERROR
14759                                                                       ;PC WAS INCORRECT
14760                                 ;
14761                                 ;COMMON CODE FOR TRAP AND NO TRAP
14762                                 ;
14763 075206  170203                  2$:   STFPS   R3                      ;SAVE FPS
14764 075210  012702  000200                MOV     #200,R2                 ;SET FPP TO DOUBLE
14765 075214  170102                        LDFPS   R2
14766 075216  012701  003142                MOV     #RECDST,R1              ;POINT TO RECEIVED DATA TABLE
14767                                 ;SAVE FRACTIONAL RESULT
14768 075222  174011                        STD     AC0,(R1)                ;SAVE AC0  RESULT
14769 075224  026503  000042                CMP     42(R5),R3               ;VERIFY STATUS
14770 075230  001403                        BEQ     3$                      ;BRANCH IF GOOD
14771 075232  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14772 075236  104003                        ERROR   .+3                     ;FPP ERROR
14773                                                                       ;BAD FPS
14774 075240  010504                  3$:   MOV     R5,R4                   ;POINT TO EXPECTED DATA
14775 075242  062704  000020                ADD     #20,R4
14776 075246  004767  042626          4$:   JSR     R7,DATVER                       ;VERIFY DATA
14777 075252  005767  105642                TST     COUNT
14778 075256  001403                        BEQ     5$                      ;BRANCH IF GOOD
14779 075260  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14780 075264  104003                        ERROR   .+3                     ;FPP ERROR
14781                                                                       ;BAD AC0
14782                                 ;SAVE INTEGER RESULT
14783                                 ;
14784 075266  174111                  5$:   STD     AC1,(R1)                ;SAVE AC1 RESULT
14785 075270  010504                        MOV     R5,R4                   ;POINT TO EXPECTED
14786 075272  062704  000030                ADD     #30,R4
14787 075276  004767  042576                JSR     R7,DATVER               ;VERIFY DATA
14788 075302  005767  105612                TST     COUNT                   ;
14789 075306  001403                        BEQ     6$                      ;BRANCH IF GOOD
14790 075310  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14791 075314  104003                        ERROR   .+3                     ;FPP ERROR
14792                                                                       ;BAD AC1
14793 075316  005737  003030          6$:   TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
14794 075322  001002                        BNE     7$                      ;BRANCH IF NEED TO CHECK
14795 075324  000165  000044                JMP     44(R5)                  ;RETURN FROM TEST
14796 075330  170301                  7$:   STST    R1                      ;SAVE FEC
```

FLOATING POINT TESTS

```
14797 075332  016504  000044                  MOV     44(R5),R4               ;GET FEC
14798 075336  020401                          CMP     R4,R1                   ;VERIFY FEC
14799 075340  001403                          BEQ     8$                      ;BRANCH IF GOOD
14800 075342  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14801 075346  104003                          ERROR   +3              ;FPP ERROR
14802                                                                 ;BAD FEC
14803 075350  000165  000046          8$:     JMP     46(R5)          ;RETURN FROM TEST
14804                                         ;
14805 075354                                  HOP15:
14808                                         ;
14809                                         ;---------------------------------
14810                                         ;TEST STCFD
14811                                         ;
14812 075354                                  MSFD:
14813                                         ;
14814                                         ;1/AC=0
14815 075354  004767  000170                          JSR     R7,SFDSUB               ;DO TEST
14816 075360  000177  000000  000000                  .WORD   0177,0,0,1              ;ACO
14817 075370  000000  000000  000000                  .WORD   0,0,0,0         ;RESULT
14818 075400  047557                                  .WORD   47557           ; TEST FPS
14819 075402  047544                                  .WORD   47544           ;RESULT FPS
14820                                         ;2/AC>0, TRUNCATE
14821 075404  004767  000140                          JSR     R7,SFDSUB       ;DO TEST
14822 075410  077577  177777  177777                  .WORD   77577,-1,-1,-1          ;ACO
14823 075420  077577  177777  000000                  .WORD   77577,-1,0,0            ;RESULT
14824 075430  007540                                  .WORD   7540            ; TEST FPS
14825 075432  007540                                  .WORD   7540            ;RESULT FPS
14826                                         ;3/AC<0, ROUND
14827 075434  004767  000110                          JSR     R7,SFDSUB               ;DO TEST
14828 075440  100377  177777  100000                  .WORD   100377,-1,100000,0              ;ACO
14829 075450  100377  177777  000000                  .WORD   100377,-1,0,0   ;RESULT
14830 075460  007517                                  .WORD   7517            ; TEST FPS
14831 075462  007510                                  .WORD   7510            ;RESULT FPS
14832                                         ;4/AC=-0
14833 075464  004767  000060                          JSR     R7,SFDSUB               ;DO TEST
14834 075470  100000  000000  000000                  .WORD   100000,0,0,0            ;ACO
14835 075500  000000  000000  000000                  .WORD   0,0,0,0         ;RESULT
14836 075510  007757                                  .WORD   7757            ; TEST FPS
14837 075512  007744                                  .WORD   7744            ;RESULT FPS
14838                                         ;5/AC<0
14839 075514  004767  000030                          JSR     R7,SFDSUB               ;DO TEST
14840 075520  125252  125252  125252                  .WORD   125252,125252,125252,125252             ;ACO
14841 075530  125252  125252  000000                  .WORD   125252,125252,0,0                       ;RESULT
14842 075540  000000                                  .WORD   0               ; TEST FPS
14843 075542  000010                                  .WORD   10              ;RESULT FPS
14844                                         ;
14845 075544  000167  000120                          JMP     HOP16                   ;GET OVER SUBROUTINE
14846                                         ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14847                                         ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14848                                         ;STCFD
14849                                         ;               ACO
14850                                         ;               RESULT
14851                                         ;               FPS BEFORE EXECUTION
14852                                         ;               FPS AFTER EXECUTION
14853                                         ;
14854                                         ;THERE CAN BE NO TRAPS WITH THE STCFD INSTRUCTION
14855                                         ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
```

FLOATING POINT TESTS

```
14856                                   ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14857                                   ;
14858 075550  012605                    SFDSUB: MOV     (SP)+,R5                ; RETURN ADDRESS TO USE AS POINTER
14859 075552  012737  075656  000244            MOV     #50$,@#FPVEC            ;REDIRECT TRAP VECTOR
14860 075560  012702  000200                    MOV     #200,R2                 ;SET TO DOUBLE MODE FOR LOAD
14861 075564  170102                            LDFPS   R2                      ;LOAD FPS
14862 075566  172415                            LDD     (R5),AC0                ;LOAD AC0 WITH TEST DATA
14863 075570  012701  003142                    MOV     #RECDST,R1              ;POINT TO RESULT AREA
14864 075574  016502  000020                    MOV     20(R5),R2               ;GET TEST FPS
14865 075600  170102                            LDFPS   R2                      ;LOAD TEST FPS
14866                                   ;
14867 075602  176011                    40$:    STCFD   AC0,(R1)                ;*TEST INSTRUCTION
14868                                   ;
14869                                   ;INSTRUCTION DIDNT TRAP
14870                                   ;VERIFY STATUS
14871                                   ;
14872 075604  170203                    2$:     STFPS   R3                      ;SAVE FPS
14873 075606  016502  000022                    MOV     22(R5),R2                       ;GET EXPECTED STATUS
14874 075612  020203                            CMP     R2,R3                   ;VERIFY STATUS
14875 075614  001403                            BEQ     3$                      ;BRANCH IF GOOD
14876 075616  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14877 075622  104003                            ERROR   +3              ;FPP ERROR
14878                                           ;BAD FPS
14879 075624  010504                    3$:     MOV     R5,R4                   ;POINT TO EXPECTED DATA
14880 075626  062704  000010                    ADD     #10,R4
14881 075632  004767  042242            4$:     JSR     R7,DATVER                       ;VERIFY DATA
14882 075636  005767  105256                    TST     COUNT
14883 075642  001403                            BEQ     5$                      ;BRANCH IF GOOD
14884 075644  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14885 075650  104003                            ERROR   +3              ;FPP ERROR
14886                                           ;BAD AC0
14887 075652  000165  000024            5$:     JMP     24(R5)                  ;RETURN FROM TEST
14888                                   ;
14889                                   ;INSTRUCTION TRAPPED
14890                                   ;
14891 075656  004737  140132            50$:    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14892 075662  104003                            ERROR   +3              ;FPP ERROR
14893                                           ;INSTRUCTION WASNT SUPPOSE TO TRAP
14894 075664  000165  000024                    JMP     24(R5)          ;RETURN FROM TEST
14895                                   ;
14896 075670                            HOP16:
14899                                   ;
14900                                   ;---------------------------------
14901                                   ;TEST STCDF
14902                                   ;
14903 075670                            MSDF:
14904                                   ;
14905                                   ;1/AC=0
14906 075670  005037  003030                    CLR     @#FLAG          ;NO INTERRUPT
14907 075674  004767  000220                    JSR     R7,SDFSUB               ;DO TEST
14908 075700  000177  000000  000000            .WORD   177,0,0,0               ;AC0
14909 075710  000000  000000                    .WORD   0,0             ;RESULT
14910 075714  000200                            .WORD   200             ; TEST FPS
14911 075716  000204                            .WORD   204             ;RESULT FPS
14912                                   ;2/AC=-0
14913 075720  005037  003030                    CLR     @#FLAG          ;NO INTERRUPT
14914 075724  004767  000170                    JSR     R7,SDFSUB               ;DO TEST
```

FLOATING POINT TESTS

```
14915 075730  100000  000300  000200          .WORD   100000,300,200,100                ;AC0
14916 075740  000000  000000                  .WORD   0,0                 ;RESULT
14917 075744  007777                          .WORD   7777                ; TEST FPS
14918 075746  007744                          .WORD   7744                ;RESULT FPS
14919                               ;3/AC>0, TRUNCATE
14920 075750  005037  003030                  CLR     @#FLAG              ;NO INTERRUPT
14921 075754  004767  000140                  JSR     R7,SDFSUB                   ;DO TEST
14922 075760  055555  055555  177777          .WORD   55555,55555,-1,-1                   ;AC0
14923 075770  055555  055555                  .WORD   55555,55555                 ;RESULT
14924 075774  000240                          .WORD   240                 ; TEST FPS
14925 075776  000240                          .WORD   240                 ;RESULT FPS
14926                               ;4/AC<0, ROUND TO UNDEFINED VARIABLE
14927 076000  012737  000001  003030          MOV     #1,@#FLAG                   ;INTERRUPT
14928 076006  004767  000106                  JSR     R7,SDFSUB                   ;DO TEST
14929 076012  077777  177777  100000          .WORD   77777,-1,100000,0                   ;AC0
14930 076022  000000  000000                  .WORD   0,0                 ;RESULT
14931 076026  001200                          .WORD   1200                ; TEST FPS
14932 076030  101206                          .WORD   101206              ;RESULT FPS
14933                               ;5/AC<0, ROUND
14934 076032  005037  003030                  CLR     @#FLAG              ;NO INTERRUPT
14935 076036  004767  000056                  JSR     R7,SDFSUB                   ;DO TEST
14936 076042  125252  125252  125252          .WORD   125252,125252,125252,125252                 ;AC0
14937 076052  125252  125253                  .WORD   125252,125253               ;RESULT
14938 076056  007700                          .WORD   7700                ; TEST FPS
14939 076060  007710                          .WORD   7710                ;RESULT FPS
14940                               ;6/ROUND TO UV, TRAPS DISABLED
14941 076062  012737  000002  003030          MOV     #2,@#FLAG                   ;INTERRUPT
14942 076070  004767  000024                  JSR     R7,SDFSUB                   ;DO TEST
14943 076074  077777  177777  177777          .WORD   77777,-1,-1,0               ;AC0
14944 076104  000000  000000                  .WORD   0,0                 ;RESULT
14945 076110  006700                          .WORD   6700                ; TEST FPS
14946 076112  006706                          .WORD   6706                ;RESULT FPS
14947                               ;
14948 076114  000167  000232                  JMP     HOP17               ;GET OVER SUBROUTINE
14949                               ;
14950                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14951                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14952                               ;STCDF
14953                               ;               AC0
14954                               ;               RESULT
14955                               ;               FPS BEFORE EXECUTION
14956                               ;               FPS AFTER EXECUTION
14957                               ;
14958                               ;A TRAP CAN ONLY OCCUR IF ROUNDING CAUSES OVERFLOW
14959                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14960                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
14961                               ;
14962 076120  012605                SDFSUB: MOV     (SP)+,R5                    ; RETURN ADDRESS TO USE AS POINTER
14963 076122  012737  076202  000244          MOV     #50$,@#FPVEC                ;REDIRECT TRAP VECTOR
14964 076130  012702  000200                  MOV     #200,R2                     ;SET TO DOUBLE MODE FOR LOAD
14965 076134  170102                          LDFPS   R2                          ;LOAD FPS
14966 076136  172415                          LDD     (R5),AC0                    ;LOAD AC0 WITH TEST DATA
14967 076140  012701  003142                  MOV     #RECDST,R1                  ;POINT TO RESULT AREA
14968 076144  016502  000014                  MOV     14(R5),R2                   ;GET TEST FPS
14969 076150  170102                          LDFPS   R2                          ;LOAD TEST FPS
14970                               ;
14971 076152  176011                40$:    STCDF   AC0,(R1)                    ;*TEST INSTRUCTION
```

```
14972 076154  170327                  1$:     STST    (PC)+                   ;WAIT FOR POSSIBLE FPA TRAP.
14973 076156  000000                          .WORD   0                       ;STORE STATUS HERE.
14974                                 ;
14975                                 ;INSTRUCTION DIDNT TRAP
14976                                 ;
14977 076160  032737  000001  003030          BIT     #1,@#FLAG                       ;VERIFY A NO TRAP CONDITION
14978 076166  001426                          BEQ     2$                      ;BRANCH IF GOOD
14979 076170  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14980 076174  104003                          ERROR   +3                    ;FPP ERROR
14981                                                                       ;INSTRUCTION SHOULD HAVE TRAPPED
14982 076176  000167  000042                  JMP     2$                      ;REJOIN CODE
14983                                 ;
14984                                 ;INSTRUCTION TRAPPED
14985                                 ;
14986 076202  032737  000001  003030  50$:     BIT     #1,@#FLAG               ;SEE IF EXPECTING A TRAP
14987 076210  001005                          BNE     51$                     ;BRANCH IF EXPECTING A TRAP
14988 076212  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14989 076216  104003                          ERROR   +3                    ;FPP ERROR
14990                                                                       ;INSTRUCTION WASNT SUPPOSE TO TRAP
14991 076220  000167  000020                  JMP     2$                      ;REJOIN CODE
14992 076224  012604                  51$:    MOV     (SP)+,R4                ;SEE IF PC = INSTRUCTION
14993 076226  005726                          TST     (SP)+                   ;CLEAN UP STACK
14994 076230  022704  076154                  CMP     #1$,R4                  ;
14995 076234  001403                          BEQ     2$                      ;BRANCH IF GOOD COMPARE
14996 076236  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
14997 076242  104003                          ERROR   +3                    ;FPP ERROR
14998                                                                       ;PC WAS INCORRECT
14999                                 ;
15000                                 ;COMMON CODE FOR TRAP AND NO TRAP
15001                                 ;VERIFY STATUS
15002                                 ;
15003 076244  170203                  2$:     STFPS   R3                      ;SAVE FPS
15004 076246  016502  000016                  MOV     16(R5),R2                       ;GET EXPECTED STATUS
15005 076252  020203                          CMP     R2,R3                   ;VERIFY STATUS
15006 076254  001403                          BEQ     3$                      ;BRANCH IF GOOD
15007 076256  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15008 076262  104003                          ERROR   +3                    ;FPP ERROR
15009                                                                       ;BAD FPS
15010 076264  010504                  3$:     MOV     R5,R4                   ;POINT TO EXPECTED DATA
15011 076266  062704  000010                  ADD     #10,R4
15012 076272  004767  041564          4$:     JSR     R7,DATVFR                       ;VERIFY DATA
15013 076276  005767  104616                  TST     COUNT
15014 076302  001403                          BEQ     5$                      ;BRANCH IF GOOD
15015 076304  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15016 076310  104003                          ERROR   +3                    ;FPP ERROR
15017                                                                       ;BAD ACO
15018 076312  005737  003030  5$:     TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
15019 076316  001002                          BNE     7$                      ;BRANCH IF NEED TO CHECK
15020 076320  000165  000020                  JMP     20(R5)                  ;RETURN FROM TEST
15021                                 ;
15022                                 ;VERIFY FEC
15023                                 ;
15024 076324  012704  003122  7$:     MOV     #RECFEC,R4              ;POINT TO FEC AREA
15025 076330  170314                  STST    (R4)                    ;SAVE FEC
15026 076332  021427  000010                  CMP     (R4),#10                ;VERIFY FEC FOR OVERFLOW
15027 076336  001403                          BEQ     8$                      ;BRANCH IF GOOD
15028 076340  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
```

FLOATING POINT TESTS

```
15029 076344  104003                    ERROR   +3              ;FPP ERROR
15030                                                             ;BAD FEC
15031 076346  000165  000020      8$:     JMP     20(R5)          ;RETURN FROM TEST
15032                             ;
15033 076352                      HOP17:
15036                             ;
15037                             ;---------------------------------
15038                             ;TEST STCFD - USING ILLEGAL ACCUMULATOR
15039                             ;
15040 076352                      MSFDI:
15041                             ;
15042 076352  012701  040000              MOV     #40000,R1               ;DISABLE INTERRUPTS
15043 076356  170101                      LDFPS   R1                      ;
15044 076360  176006                      STCFD   AC0,AC6                 ;*TEST ILLEGAL INSTRUCTION
15045 076362  170202                      STFPS   R2                      ;SAVE STATUS
15046 076364  170303                      STST    R3                      ;SAVE FEC
15047 076366  022702  140000              CMP     #140000,R2              ;VERIFY FER SET
15048 076372  001403                      BEQ     1$                      ;BRANCH IF ERROR RECEIVED
15049 076374  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15050 076400  104003                      ERROR   +3              ;FPP ERROR
15051                                                             ;FER BIT NOT SET ON ILLEGAL INST.
15052 076402  022703  000002      1$:     CMP     #2,R3                   ;VERIFY FEC = FLOATING OPCDOE ERROR
15053 076406  001403                      BEQ     2$                      ;BRANCH IF GOOD
15054 076410  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15055 076414  104003                      ERROR   +3              ;FPP ERROR
15056                                                             ;FEC INCORRECT
15057 076416                      2$:
15058                             ;
15061                             ;
15062                             ;---------------------------------
15063                             ;TEST CLRD
15064                             ;
15065 076416                      MCLRD:
15066                             ;
15067 076416  012701  003764              MOV     #TAB47,R1               ;POINT TO DATA
15068 076422  012704  000200              MOV     #200,R4                 ;SET FPP STATUS TO DOUBLE
15069 076426  170104                      LDFPS   R4
15070 076430  172411                      LDD     (R1),AC0                ;
15071 076432  012701  003142              MOV     #RECDST,R1              ;POINT TO DATA BUFFER
15072 076436  174011                      STD     AC0,(R1)                ;STORE GARBAGE
15073 076440  170411                      CLRD    (R1)                    ;CLEAR DATA BUFFER
15074 076442  012704  003314              MOV     #TAB6,R4                ;VERIFY BUFFER =0
15075 076446  004767  041426              JSR     R7,DATVER               ;
15076 076452  005767  104442              TST     COUNT                   ;
15077 076456  001403                      BEQ     1$                      ;BRANCH I RECDST = 0
15078 076460  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15079 076464  104003                      ERROR   +3              ;FPP ERROR
15080                                                             ;RECDST NOT CLEARED
15081 076466  170202              1$:     STFPS   R2                      ;SAVE STATUS
15082 076470  020227  000204              CMP     R2,#204                 ;VERIFY STATUS
15083 076474  001403                      BEQ     2$                      ;BRANCH IF GOOD
15084 076476  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15085 076502  104003                      ERROR   +3              ;FPP ERROR
15086                                                             ;BAD STATUS
15087 076504                      2$:
15090                             ;
15091                             ;---------------------------------
```

FLOATING POINT TESTS

```
15092                                   ;TEST CLRD, ILLEGAL ACCUMULATOR
15093                                   ;
15094 076504                            MCLRI:
15095                                   ;
15096 076504  012704  040200                    MOV     #40200,R4               ;DISABLE INTERRUPTS
15097 076510  170104                            LDFPS   R4                      ;LOAD STATUS
15098 076512  170406                            CLRD    R6                      ;*TEST INSTRUCTION WITH ILLEGAL ACC
15099 076514  170203                            STFPS   R3                      ;SAVE STATUS
15100 076516  170305                            STST    R5                      ;SAVE FEC
15101 076520  022703  140200                    CMP     #140200,R3              ;VERIFY ERROR
15102 076524  001403                            BEQ     1$                      ;BRANCH IF FER SET
15103 076526  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15104 076532  104003                            ERROR   .+3                 ;FPP ERROR
15105                                                                       ;ERROR IN FPS
15106 076534  022705  000002            1$:     CMP     #2,R5               ;VERIFY FEC =2 OPCODE ERROR
15107 076540  001403                            BEQ     2$                  ;BRANCH IF GOOD
15108 076542  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15109 076546  104003                            ERROR   .+3                 ;FPP ERROR
15110                                                                           ;BAD FEC
15111 076550                            2$:
15114                                   ;
15115                                   ;---------------------------------
15116                                   ;TEST LDFPS, STFPS MODE 1
15117                                   ;
15118 076550                            MLS1:
15119                                   ;
15120 076550  012704  003162                    MOV     #TSTLOC,R4              ;POINT R4 TO RAM
15121 076554  012714  147757                    MOV     #147757,(R4)            ;SETUP EXPECTED STATUS
15122 076560  012701  003132                    MOV     #RECST,R1               ;SET BUFFER FOR RECEIVED STATUS
15123 076564  012737  076650  000244            MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
15124 076572  170114                            LDFPS   (R4)                    ;*TEST INSTRUCTION
15125 076574  170211                            STFPS   (R1)                    ;*TEST INSTRUCTION
15126 076576  020427  003162                    CMP     R4,#TSTLOC              ;VERIFY R4
15127 076602  001403                            BEQ     1$                      ;BRANCH IF GOOD
15128 076604  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15129 076610  104003                            ERROR   .+3                 ;FPP ERROR
15130                                                                       ;
15131 076612  020127  003132            1$:     CMP     R1,#RECST               ;VERIFY R1
15132 076616  001403                            BEQ     2$                      ;BRANCH IF GOOD
15133 076620  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15134 076624  104003                            ERROR   .+3                 ;FPP ERROR
15135                                                                       ;BAD R1
15136 076626  023727  003132  147757    2$:     CMP     @#RECST,#147757         ;VERIFY STATUS
15137 076634  001412                            BEQ     3$                      ;BRANCH F GOOD
15138 076636  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15139 076642  104003                            ERROR   .+3                 ;FPP ERROR
15140                                                                       ;BAD STATUS
15141 076644  000167  000012                    JMP     3$                      ;GET OVER TRAP
15142                                   ;
15143                                   ;UNEXPECTED TRAP
15144                                   ;
15145 076650  012600                    10$:    MOV     (SP)+,R0                        ;SAVE PC
15146 076652  012605                            MOV     (SP)+,R5                        ;SAVE PS
15147 076654  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15148 076660  104003                            ERROR   .+3                 ;FPP ERROR
15149                                                                       ;UNEXPECTED TRAP
15150 076662                            3$:
```

```
15153                                   ;
15154                                   ;---------------------------------
15155                                   ;TEST LDFPS, STFPS MODE 2
15156                                   ;
15157 076662                            MLS2:
15158                                   ;
15159 076662  012704  003162                      MOV     #TSTLOC,R4              ;POINT R4 TO RAM
15160 076666  012714  145557                      MOV     #145557,(R4)            ;SETUP EXPECTED STATUS
15161 076672  012701  003132                      MOV     #RECST,R1               ;SET BUFFER FOR RECEIVED STATUS
15162 076676  012737  076762  000244              MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
15163 076704  170124                              LDFPS   (R4)+                   ;*TEST INSTRUCTION
15164 076706  170221                              STFPS   (R1)+                   ;*TEST INSTRUCTION
15165 076710  020427  003164                      CMP     R4,#TSTLOC+2            ;VERIFY R4
15166 076714  001403                              BEQ     1$                      ;BRANCH IF GOOD
15167 076716  004737  140132                      CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15168 076722  104003                              ERROR   +3                      ;FPP ERROR
15169                                                                             ;
15170 076724  020127  003134            1$:       CMP     R1,#RECST+2             ;VERIFY R1
15171 076730  001403                              BEQ     2$                      ;BRANCH IF GOOD
15172 076732  004737  140132                      CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15173 076736  104003                              ERROR   +3                      ;FPP ERROR
15174                                                                             ;BAD R1
15175 076740  023727  003132  145557    2$:       CMP     @#RECST,#145557         ;VERIFY STATUS
15176 076746  001412                              BEQ     3$                      ;BRANCH F GOOD
15177 076750  004737  140132                      CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15178 076754  104003                              ERROR   +3                      ;FPP ERROR
15179                                                                             ;BAD STATUS
15180 076756  000167  000012                      JMP     3$                      ;GET OVER TRAP
15181                                   ;
15182                                   ;UNEXPECTED TRAP
15183                                   ;
15184 076762  012600                    10$:      MOV     (SP)+,R0                        ;SAVE PC
15185 076764  012605                              MOV     (SP)+,R5                        ;SAVE PS
15186 076766  004737  140132                      CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15187 076772  104003                              ERROR   +3                      ;FPP ERROR
15188                                                                             ;UNEXPECTED TRAP
15189 076774                            3$:
15192                                   ;
15193                                   ;---------------------------------
15194                                   ;TEST LDFPS, STFPS MODE 3
15195                                   ;
15196 076774                            MLS3:
15197                                   ;
15198 076774  012704  003162                      MOV     #TSTLOC,R4              ;POINT R4 TO RAM
15199 077000  012737  003166  003162              MOV     #TSTLOC+4,@#TSTLOC      ;TSTLOC= DEFERRED ADDRESS
15200 077006  012737  147501  003166              MOV     #147501,@#TSTLOC+4      ;SETUP EXPECTED STATUS
15201 077014  012701  003172                      MOV     #TSTLOC+10,R1           ;R1 POINTS TO TSTLOC+10
15202 077020  012737  003132  003172              MOV     #RECST,@#TSTLOC+10      ;SET DEFERRED BUFFER FOR RECEIVED STATUS
15203 077026  012737  077112  000244              MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
15204 077034  170134                              LDFPS   @(R4)+                  ;*TEST INSTRUCTION
15205 077036  170231                              STFPS   @(R1)+                  ;*TEST INSTRUCTION
15206 077040  020427  003164                      CMP     R4,#TSTLOC+2            ;VERIFY R4
15207 077044  001403                              BEQ     1$                      ;BRANCH IF GOOD
15208 077046  004737  140132                      CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15209 077052  104003                              ERROR   +3                      ;FPP ERROR
15210                                                                             ;
15211 077054  020127  003174            1$:       CMP     R1,#TSTLOC+12           ;VERIFY R1
```

FLOATING POINT TESTS

```
15212 077060  001403                        BEQ     2$                      ;BRANCH IF GOOD
15213 077062  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15214 077066  104003                        ERROR   +3                      ;FPP ERROR
15215                                                                       ;BAD R1
15216 077070  023727  003132  147501  2$:   CMP     @#RECST,#147501         ;VERIFY STATUS
15217 077076  001412                        BEQ     3$                      ;BRANCH F GOOD
15218 077100  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15219 077104  104003                        ERROR   +3                      ;FPP ERROR
15220                                                                       ;BAD STATUS
15221 077106  000167  000012                JMP     3$                      ;GET OVER TRAP
15222                               ;
15223                               ;UNEXPECTED TRAP
15224                               ;
15225 077112  012600                10$:    MOV     (SP)+,R0                        ;SAVE PC
15226 077114  012605                        MOV     (SP)+,R5                        ;SAVE PS
15227 077116  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15228 077122  104003                        ERROR   +3                      ;FPP ERROR
15229                                                                       ;UNEXPECTED TRAP
15230 077124                        3$:
15233                               ;
15234                               ;---------------------------------
15235                               ;TEST LDFPS, STFPS MODE 4
15236                               ;
15237 077124                        MLS4:
15238                               ;
15239 077124  012704  003164                MOV     #TSTLOC+2,R4            ;POINT R4 TO RAM
15240 077130  012737  147757  003162        MOV     #147757,@#TSTLOC        ;TSTLOC= STATUS ADDRESS
15241 077136  012701  003134                MOV     #RECST+2,R1             ;SET BUFFER FOR RECEIVED STATUS
15242 077142  012737  077226  000244        MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
15243 077150  170144                        LDFPS   -(R4)                   ;*TEST INSTRUCTION
15244 077152  170241                        STFPS   -(R1)                   ;*TEST INSTRUCTION
15245 077154  020427  003162                CMP     R4,#TSTLOC              ;VERIFY R4
15246 077160  001403                        BEQ     1$                      ;BRANCH IF GOOD
15247 077162  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15248 077166  104003                        ERROR   +3                      ;FPP ERROR
15249                                                                       ;
15250 077170  020127  003132          1$:   CMP     R1,#RECST               ;VERIFY R1
15251 077174  001403                        BEQ     2$                      ;BRANCH IF GOOD
15252 077176  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15253 077202  104003                        ERROR   +3                      ;FPP ERROR
15254                                                                       ;BAD R1
15255 077204  023727  003132  147757  2$:   CMP     @#RECST,#147757         ;VERIFY STATUS
15256 077212  001412                        BEQ     3$                      ;BRANCH F GOOD
15257 077214  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15258 077220  104003                        ERROR   +3                      ;FPP ERROR
15259                                                                       ;BAD STATUS
15260 077222  000167  000012                JMP     3$                      ;GET OVER TRAP
15261                               ;
15262                               ;UNEXPECTED TRAP
15263                               ;
15264 077226  012600                10$:    MOV     (SP)+,R0                        ;SAVE PC
15265 077230  012605                        MOV     (SP)+,R5                        ;SAVE PS
15266 077232  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15267 077236  104003                        ERROR   +3                      ;FPP ERROR
15268                                                                       ;UNEXPECTED TRAP
15269 077240                        3$:
15272                               ;
```

```
15273                                   ;---------------------------------
15274                                   ;TEST LDFPS, STFPS MODE 5
15275                                   ;
15276 077240                            MLS5:
15277                                   ;
15278 077240  012704  003164                    MOV     #TSTLOC+2,R4            ;POINT R4 TO RAM
15279 077244  012737  003166  003162      MOV     #TSTLOC+4,@#TSTLOC      ;TSTLOC= DEFERRED ADDRESS
15280 077252  012737  147501  003166      MOV     #147501,@#TSTLOC+4      ;SETUP EXPECTED STATUS
15281 077260  012701  003174              MOV     #TSTLOC+12,R1           ;R1 POINTS TO 412
15282 077264  012737  003132  003172      MOV     #RECST,@#TSTLOC+10      ;SET DEFERRED BUFFER FOR RECEIVED STATUS
15283 077272  012737  077356  000244      MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
15284 077300  170154                      LDFPS   @-(R4)                  ;*TEST INSTRUCTION
15285 077302  170251                      STFPS   @-(R1)                  ;*TEST INSTRUCTION
15286 077304  020427  003162              CMP     R4,#TSTLOC              ;VERIFY R4
15287 077310  001403                      BEQ     1$                      ;BRANCH IF GOOD
15288 077312  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15289 077316  104003                      ERROR   +3              ;FPP ERROR
15290                                                           ;
15291 077320  020127  003172      1$:     CMP     R1,#TSTLOC+10           ;VERIFY R1
15292 077324  001403                      BEQ     2$                      ;BRANCH IF GOOD
15293 077326  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15294 077332  104003                      ERROR   +3              ;FPP ERROR
15295                                                           ;BAD R1
15296 077334  023727  003132  147501  2$: CMP     @#RECST,#147501         ;VERIFY STATUS
15297 077342  001412                      BEQ     3$                      ;BRANCH F GOOD
15298 077344  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15299 077350  104003                      ERROR   +3              ;FPP ERROR
15300                                                           ;BAD STATUS
15301 077352  000167  000012              JMP     3$                      ;GET OVER TRAP
15302                                   ;
15303                                   ;UNEXPECTED TRAP
15304                                   ;
15305 077356  012600                10$:     MOV     (SP)+,R0                        ;SAVE PC
15306 077360  012605                      MOV     (SP)+,R5                        ;SAVE PS
15307 077362  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15308 077366  104003                      ERROR   +3              ;FPP ERROR
15309                                                           ;UNEXPECTED TRAP
15310 077370                            3$:
15313                                   ;
15314                                   ;---------------------------------
15315                                   ;TEST LDFPS, STFPS MODE 6
15316                                   ;
15317 077370                            MLS6:
15318                                   ;
15319 077370  012704  003162                    MOV     #TSTLOC,R4              ;POINT R4 TO RAM
15320 077374  012737  140001  003166      MOV     #140001,@#TSTLOC+4      ;SETUP EXPECTED STATUS
15321 077402  012701  003272              MOV     #TSTLOC+110,R1          ;R1 WILL POINT TO TESTLOC+10
15322 077406  012737  077476  000244      MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
15323 077414  170164  000004              LDFPS   4(R4)                   ;*TEST INSTRUCTION
15324 077420  170261  177700              STFPS   -100(R1)                        ;*TEST INSTRUCTION
15325 077424  020427  003162              CMP     R4,#TSTLOC              ;VERIFY R4
15326 077430  001403                      BEQ     1$                      ;BRANCH IF GOOD
15327 077432  004737  140132              CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15328 077436  104003                      ERROR   +3              ;FPP ERROR
15329                                                           ;
15330 077440  020127  003272      1$:     CMP     R1,#TSTLOC+110          ;VERIFY R1
15331 077444  001403                      BEQ     2$                      ;BRANCH IF GOOD
```

FLOATING POINT TESTS

```
15332 077446  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15333 077452  104003                          ERROR   +3              ;FPP ERROR
15334                                                                 ;BAD R1
15335 077454  023727  003172  140001  2$:     CMP     @#TSTLOC+10,#140001     ;VERIFY STATUS
15336 077462  001412                          BEQ     3$                      ;BRANCH F GOOD
15337 077464  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15338 077470  104003                          ERROR   +3              ;FPP ERROR
15339                                                                 ;BAD STATUS
15340 077472  000167  000012                  JMP     3$                      ;GET OVER TRAP
15341                                 ;
15342                                 ;UNEXPECTED TRAP
15343                                 ;
15344 077476  012600                  10$:    MOV     (SP)+,R0                        ;SAVE PC
15345 077500  012605                          MOV     (SP)+,R5                        ;SAVE PS
15346 077502  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15347 077506  104003                          ERROR   +3              ;FPP ERROR
15348                                                                 ;UNEXPECTED TRAP
15349 077510                          3$:
15352                                 ;
15353                                 ;---------------------------------
15354                                 ;TEST LDFPS, STFPS MODE 7
15355                                 ;
15356 077510                          MLS7:
15357                                 ;
15358 077510  012704  003262                  MOV     #TSTLOC+100,R4          ;POINT R4 TO RAM
15359 077514  012737  003166  003162          MOV     #TSTLOC+4,@#TSTLOC      ;TSTLOC= DEFERRED ADDRESS
15360 077522  012737  145501  003166          MOV     #145501,@#TSTLOC+4      ;SETUP EXPECTED STATUS
15361 077530  012701  003072                  MOV     #TSTLOC-70,R1           ;R1 POINTS TO TSTLOC+10
15362 077534  012737  003172  003164          MOV     #TSTLOC+10,@#TSTLOC+2   ;
15363 077542  012737  077632  000244          MOV     #10$,@#FPVEC            ;SETUP TRAP VECTOR
15364 077550  170174  177700                  LDFPS   @-100(R4)                       ;*TEST INSTRUCTION
15365 077554  170271  000072                  STFPS   @72(R1)                 ;*TEST INSTRUCTION
15366 077560  020427  003262                  CMP     R4,#TSTLOC+100          ;VERIFY R4
15367 077564  001403                          BEQ     1$                      ;BRANCH IF GOOD
15368 077566  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15369 077572  104003                          ERROR   +3              ;FPP ERROR
15370                                                                 ;
15371 077574  020127  003072          1$:     CMP     R1,#TSTLOC-70           ;VERIFY R1
15372 077600  001403                          BEQ     2$                      ;BRANCH IF GOOD
15373 077602  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15374 077606  104003                          ERROR   +3              ;FPP ERROR
15375                                                                 ;BAD R1
15376 077610  023727  003172  145501  2$:     CMP     @#TSTLOC+10,#145501     ;VERIFY STATUS
15377 077616  001412                          BEQ     3$                      ;BRANCH F GOOD
15378 077620  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15379 077624  104003                          ERROR   +3              ;FPP ERROR
15380                                                                 ;BAD STATUS
15381 077626  000167  000012                  JMP     3$                      ;GET OVER TRAP
15382                                 ;
15383                                 ;UNEXPECTED TRAP
15384                                 ;
15385 077632  012600                  10$:    MOV     (SP)+,R0                        ;SAVE PC
15386 077634  012605                          MOV     (SP)+,R5                        ;SAVE PS
15387 077636  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15388 077642  104003                          ERROR   +3              ;FPP ERROR
15389                                                                 ;UNEXPECTED TRAP
15390 077644                          3$:
```

```
15393                                   ;
15394                                   ;---------------------------------
15395                                   ;TEST LDCLD MODE 27
15396                                   ;
15397 077644                            MLDC2:
15398                                   ;
15399 077644  005001                            CLR     R1                      ;INIT R1
15400 077646  012704  007700                      MOV     #7700,R4                ;FPS=DOUBLE, LONG
15401 077652  170104                            LDFPS   R4                      ;
15402 077654  012737  077714  000244              MOV     #10$,@#FPVEC            ;SETUP WILD TRAP
15403 077662  177027                            LDCLD   (R7)+,AC0               ;*TEST INSTRUCTION
15404 077664  005201                            INC     R1                      ;
15405 077666  005201                            INC     R1                      ;
15406 077670  005201                            INC     R1                      ;
15407 077672  005201                            INC     R1                      ;
15408 077674  020127  000003                    CMP     R1,#3                   ;VERIFY
15409 077700  001412                            BEQ     1$                      ;BRANCH IF GOOD
15410 077702  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15411 077706  104003                            ERROR   +3                      ;FPP ERROR
15412                                                                           ;INSTRUCTION FAILED
15413 077710  000167  000012                    JMP     1$                      ;JUMP OVER WILD TRAP
15414 077714  012600                    10$:    MOV     (SP)+,R0                ;SAVE PC
15415 077716  012605                            MOV     (SP)+,R5                        ;SAVE PS
15416 077720  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15417 077724  104003                            ERROR   +3                      ;FPP ERROR
15418                                                                           ;WILD TRAP ON INSTRUCTION
15419 077726  012704  003324              1$:   MOV     #TAB6A,R4               ;POINT TO EXPECTED DATA
15420 077732  012701  003142                    MOV     #RECDST,R1              ;POINT TO DATA BUFFER
15421 077736  174011                            STD     AC0,(R1)                ;VERIFY DATA
15422 077740  004767  040134                    JSR     R7,DATVER               ;
15423 077744  005767  103150                    TST     COUNT
15424 077750  001403                            BEQ     2$                      ;BRANCH IF GOOD DATA
15425 077752  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15426 077756  104003                            ERROR   +3                      ;FPP ERROR
15427                                                                           ;BAD DATA
15428 077760                            2$:
15431                                   ;
15432                                   ;---------------------------------
15433                                   ;TEST LDCIF, LDCLF
15434                                   ;
15435 077760                            MLCF:
15436                                   ;
15437                                   ;1/INT=0
15438 077760  004767  000500                    JSR     R7,LCFSUB               ;DO TEST
15439 077764  000000  000000                    .WORD   0,0                     ;FSRC
15440 077770  000000  000000                    .WORD   0,0                     ;RESULT
15441 077774  000000                            .WORD   0                       ; TEST FPS
15442 077776  000004                            .WORD   4                       ;RESULT FPS
15443                                   ;2/INT=0,-1
15444 100000  004767  000460                    JSR     R7,LCFSUB               ;DO TEST
15445 100004  000000  177777                    .WORD   0,-1                    ;FSRC
15446 100010  000000  000000                    .WORD   0,0                     ;RESULT
15447 100014  007440                            .WORD   7440                    ; TEST FPS
15448 100016  007444                            .WORD   7444                    ;RESULT FPS
15449                                   ;3/LONG=0
15450 100020  004767  000440                    JSR     R7,LCFSUB               ;DO TEST
15451 100024  000000  000000                    .WORD   0,0                     ;FSRC
```

FLOATING POINT TESTS

```
15452 100030  000000  000000                .WORD   0,0             ;RESULT
15453 100034  000100                        .WORD   100             ; TEST FPS
15454 100036  000104                        .WORD   104             ;RESULT FPS
15455                             ;4/INT=40000
15456 100040  004767  000420                JSR     R7,LCFSUB               ;DO TEST
15457 100044  040000  000000                .WORD   40000,0         ;FSRC
15458 100050  043600  000000                .WORD   43600,0         ;RESULT
15459 100054  000017                        .WORD   17              ; TEST FPS
15460 100056  000000                        .WORD   0               ;RESULT FPS
15461                             ;5/LONG=1
15462 100060  004767  000400                JSR     R7,LCFSUB               ;DO TEST
15463 100064  000000  000001                .WORD   0,1             ;FSRC
15464 100070  040200  000000                .WORD   40200,0         ;RESULT
15465 100074  000117                        .WORD   117             ; TEST FPS
15466 100076  000100                        .WORD   100             ;RESULT FPS
15467                             ;6/INT=PATTERN
15468 100100  004767  000360                JSR     R7,LCFSUB               ;DO TEST
15469 100104  000252  025252                .WORD   252,25252       ;FSRC
15470 100110  042052  000000                .WORD   42052,0         ;RESULT
15471 100114  000000                        .WORD   0               ; TEST FPS
15472 100116  000000                        .WORD   0               ;RESULT FPS
15473                             ;7/INT=-40000
15474 100120  004767  000340                JSR     R7,LCFSUB               ;DO TEST
15475 100124  140000  000000                .WORD   -40000,0        ;FSRC
15476 100130  143600  000000                .WORD   143600,0                ;RESULT
15477 100134  000007                        .WORD   7               ; TEST FPS
15478 100136  000010                        .WORD   10              ;RESULT FPS
15479                             ;8/INT=-1
15480 100140  004767  000320                JSR     R7,LCFSUB               ;DO TEST
15481 100144  177777  000000                .WORD   -1,0            ;FSRC
15482 100150  140200  000000                .WORD   140200,0                ;RESULT
15483 100154  000007                        .WORD   7               ; TEST FPS
15484 100156  000010                        .WORD   10              ;RESULT FPS
15485                             ;9/INT=PATTERN
15486 100160  004767  000300                JSR     R7,LCFSUB               ;DO TEST
15487 100164  125252  125252                .WORD   125252,125252           ;FSRC
15488 100170  143652  126000                .WORD   143652,126000           ;RESULT
15489 100174  000007                        .WORD   7               ; TEST FPS
15490 100176  000010                        .WORD   10              ;RESULT FPS
15491                             ;10/LONG=40000
15492 100200  004767  000260                JSR     R7,LCFSUB               ;DO TEST
15493 100204  040000  000000                .WORD   40000,0         ;FSRC
15494 100210  047600  000000                .WORD   47600,0         ;RESULT
15495 100214  000117                        .WORD   117             ; TEST FPS
15496 100216  000100                        .WORD   100             ;RESULT FPS
15497                             ;11/LONG=1
15498 100220  004767  000240                JSR     R7,LCFSUB               ;DO TEST
15499 100224  000000  000001                .WORD   0,1             ;FSRC
15500 100230  040200  000000                .WORD   40200,0         ;RESULT
15501 100234  007557                        .WORD   7557            ; TEST FPS
15502 100236  007540                        .WORD   7540            ;RESULT FPS
15503                             ;12/LONG=PATTERN
15504 100240  004767  000220                JSR     R7,LCFSUB               ;DO TEST
15505 100244  000000  000252                .WORD   0,252           ;FSRC
15506 100250  042052  000000                .WORD   42052,0         ;RESULT
15507 100254  007557                        .WORD   7557            ; TEST FPS
15508 100256  007540                        .WORD   7540            ;RESULT FPS
```

FLOATING POINT TESTS

```
15509
15510 100260  004767  000200          ;13/LONG = -40000
15511 100264  140000  000000                  JSR     R7,LCFSUB               ;DO TEST
15512 100270  147600  000000                  .WORD   -40000,0                ;FSRC
15513 100274  000107                          .WORD   147600,0                ;RESULT
15514 100276  000110                          .WORD   107             ; TEST FPS
15515                                         .WORD   110             ;RESULT FPS
15516 100300  004767  000160    ;14/LONG=-1
15517 100304  177777  177777            JSR     R7,LCFSUB               ;DO TEST
15518 100310  140200  000000            .WORD   -1,-1           ;FSRC
15519 100314  007500                    .WORD   140200,0                ;RESULT
15520 100316  007510                    .WORD   7500            ; TEST FPS
15521                                   .WORD   7510            ;RESULT FPS
15522 100320  004767  000140    ;15/LONG=PATTERN
15523 100324  125252  125252            JSR     R7,LCFSUB               ;DO TEST
15524 100330  147652  125253            .WORD   125252,125252           ;FSRC
15525 100334  000105                    .WORD   147652,125253           ;RESULT
15526 100336  000110                    .WORD   105             ; TEST FPS
15527                                   .WORD   110             ;RESULT FPS
15528 100340  004767  000120    ;16/LONG=77777,177500
15529 100344  077777  177500            JSR     R7,LCFSUB               ;DO TEST
15530 100350  047777  177777            .WORD   77777,177500            ;FSRC
15531 100354  000117                    .WORD   47777,177777            ;RESULT
15532 100356  000100                    .WORD   117             ; TEST FPS
15533                                   .WORD   100             ;RESULT FPS
15534 100360  004767  000100    ;17/LONG=40000,100
15535 100364  040000  000100            JSR     R7,LCFSUB               ;DO TEST
15536 100370  047600  000001            .WORD   40000,100               ;FSRC
15537 100374  007502                    .WORD   47600,1         ;RESULT
15538 100376  007500                    .WORD   7502            ; TEST FPS
15539                                   .WORD   7500            ;RESULT FPS
15540 100400  004767  000060    ;18/LONG=40000,100 - TRUNCATE
15541 100404  040000  000100            JSR     R7,LCFSUB               ;DO TEST
15542 100410  047600  000000            .WORD   40000,100               ;FSRC
15543 100414  007557                    .WORD   47600,0         ;RESULT
15544 100416  007540                    .WORD   7557            ; TEST FPS
15545                                   .WORD   7540            ;RESULT FPS
15546 100420  004767  000040    ;19/INT= MOST NEGATIVE
15547 100424  100000  000000            JSR     R7,LCFSUB               ;DO TEST
15548 100430  144000  000000            .WORD   100000,0                ;FSRC
15549 100434  000007                    .WORD   144000,0                ;RESULT
15550 100436  000010                    .WORD   7               ; TEST FPS
15551                                   .WORD   10              ;RESULT FPS
15552 100440  004767  000020    ;20/LONG= MOST NEGATIVE
15553 100444  100000  000000            JSR     R7,LCFSUB               ;DO TEST
15554 100450  150000  000000            .WORD   100000,0                ;FSRC
15555 100454  000107                    .WORD   150000,0                ;RESULT
15556 100456  000110                    .WORD   107             ; TEST FPS
15557                                   .WORD   110             ;RESULT FPS
                                ;
15558 100460  000167  000126            JMP     HOP18           ;GET OVER SUBROUTINE
15559                           ;
15560                           ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15561                           ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15562                           ;LDCIF, LDCLF
15563                           ;
15564                           ;               FSRC
15565                           ;               RESULT
```

FLOATING POINT TESTS

```
15566                                   ;               FPS BEFORE EXECUTION
15567                                   ;               FPS AFTER EXECUTION
15568                                   ;
15569                                   ;NO TRAP CAN OCCUR
15570                                   ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15571                                   ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15572                                   ;
15573 100464  012602                    LCFSUB: MOV     (SP)+,R2        ; RETURN ADDRESS TO USE AS POINTER
15574 100466  012737  100574  000244            MOV     #50$,@#FPVEC            ;REDIRECT TRAP VECTOR
15575 100474  012701  003142                    MOV     #RECDST,R1              ;POINT TO RESULT AREA
15576 100500  016200  000010                    MOV     10(R2),R0               ;GET TEST FPS
15577 100504  170100                            LDFPS   R0                      ;LOAD TEST FPS
15578 100506  010204                            MOV     R2,R4                   ;POINT TO TEST DATA
15579                                   ;
15580 100510  177014                    40$:    LDCIF   (R4),AC0                ;*TEST INSTRUCTION (ACCORDING TO MODE)
15581                                   ;
15582                                   ;VERIFY STATUS
15583                                   ;
15584 100512  170203                    2$:     STFPS   R3                      ;SAVE FPS
15585 100514  012700  000200                    MOV     #200,R0                 ;SET FPP STATUS TO DOUBLE
15586 100520  170100                            LDFPS   R0                      ;
15587 100522  174011                            STD     AC0,(R1)                ;SAVE TEST RESULT INTO RECDST
15588 100524  016200  000012                    MOV     12(R2),R0                       ;GET EXPECTED STATUS
15589 100530  020003                            CMP     R0,R3                   ;VERIFY STATUS
15590 100532  001403                            BEQ     3$                      ;BRANCH IF GOOD
15591 100534  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15592 100540  104003                            ERROR   +3              ;FPP ERROR
15593                                                                   ;BAD FPS
15594 100542  010204                    3$:     MOV     R2,R4                   ;POINT TO EXPECTED DATA
15595 100544  062704  000004                    ADD     #4,R4
15596 100550  004767  037306            4$:     JSR     R7,DATVFR                       ;VERIFY DATA
15597 100554  005767  102340                    TST     COUNT
15598 100560  001403                            BEQ     5$                      ;BRANCH IF GOOD
15599 100562  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15600 100566  104003                            ERROR   +3              ;FPP ERROR
15601                                                                   ;BAD AC0
15602 100570  000162  000014            5$:     JMP     14(R2)                  ;RETURN FROM TEST
15603                                   ;
15604                                   ;INSTRUCTION TRAPPED
15605                                   ;
15606 100574  012600                    50$:    MOV     (SP)+,R0                ;SAVE PC
15607 100576  012605                            MOV     (SP)+,R5                ;SAVE PS
15608 100600  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15609 100604  104003                            ERROR   +3              ;FPP ERROR
15610                                                                   ;INSTRUCTION WASNT SUPPOSE TO TRAP
15611 100606  000167  177756                    JMP     5$                      ;CONTINUE
15612 100612                            HOP18:
15615                                   ;
15616                                   ;---------------------------------
15617                                   ;TEST LDCID, LDCLD
15618                                   ;
15619 100612                            MLCD:
15620                                   ;
15621                                   ;1/LONG=0
15622 100612  004767  000264                    JSR     R7,LCDSUB               ;DO TEST
15623 100616  000000  000000                    .WORD   0,0             ;FSRC
15624 100622  000000  000000  000000            .WORD   0,0,0,0         ;RESULT
```

FLOATING POINT TESTS

```
15625 100632  007313                        .WORD   7313            ; TEST FPS
15626 100634  007304                        .WORD   7304            ;RESULT FPS
15627                               ;2/INT=0
15628 100636  004767  000240                JSR     R7,LCDSUB               ;DO TEST
15629 100642  000000  000001                .WORD   0,1             ;FSRC
15630 100646  040200  000000  000000        .WORD   40200,0,0,0             ;RESULT
15631 100656  007757                        .WORD   7757            ; TEST FPS
15632 100660  007740                        .WORD   7740            ;RESULT FPS
15633                               ;3/INT=40000
15634 100662  004767  000214                JSR     R7,LCDSUB               ;DO TEST
15635 100666  040000  177777                .WORD   40000,-1                ;FSRC
15636 100672  043600  000000  000000        .WORD   43600,0,0,0             ;RESULT
15637 100702  007617                        .WORD   7617            ; TEST FPS
15638 100704  007600                        .WORD   7600            ;RESULT FPS
15639                               ;4/INT=-40000
15640 100706  004767  000170                JSR     R7,LCDSUB               ;DO TEST
15641 100712  140000  177777                .WORD   -40000,-1               ;FSRC
15642 100716  143600  000000  000000        .WORD   143600,0,0,0            ;RESULT
15643 100726  007600                        .WORD   7600            ; TEST FPS
15644 100730  007610                        .WORD   7610            ;RESULT FPS
15645                               ;5/LONG=40000
15646 100732  004767  000144                JSR     R7,LCDSUB               ;DO TEST
15647 100736  040000  000000                .WORD   40000,0         ;FSRC
15648 100742  047600  000000  000000        .WORD   47600,0,0,0             ;RESULT
15649 100752  007757                        .WORD   7757            ; TEST FPS
15650 100754  007740                        .WORD   7740            ;RESULT FPS
15651                               ;6/LONG=1
15652 100756  004767  000120                JSR     R7,LCDSUB               ;DO TEST
15653 100762  000000  000001                .WORD   0,1             ;FSRC
15654 100766  040200  000000  000000        .WORD   40200,0,0,0             ;RESULT
15655 100776  000300                        .WORD   300             ; TEST FPS
15656 101000  000300                        .WORD   300             ;RESULT FPS
15657                               ;7/LONG=-2
15658 101002  004767  000074                JSR     R7,LCDSUB               ;DO TEST
15659 101006  177777  177776                .WORD   -1,-2           ;FSRC
15660 101012  140400  000000  000000        .WORD   140400,0,0,0            ;RESULT
15661 101022  007300                        .WORD   7300            ; TEST FPS
15662 101024  007310                        .WORD   7310            ;RESULT FPS
15663                               ;8/INT=PATTERN
15664 101026  004767  000050                JSR     R7,LCDSUB               ;DO TEST
15665 101032  123456  176543                .WORD   123456,176543           ;FSRC
15666 101036  143661  122000  000000        .WORD   143661,122000,0,0               ;RESULT
15667 101046  000200                        .WORD   200             ; TEST FPS
15668 101050  000210                        .WORD   210             ;RESULT FPS
15669                               ;9/LONG=PATTERN
15670 101052  004767  000024                JSR     R7,LCDSUB               ;DO TEST
15671 101056  125252  125252                .WORD   125252,125252           ;FSRC
15672 101062  147652  125252  126000        .WORD   147652,125252,126000,0          ;RESULT
15673 101072  000300                        .WORD   300             ; TEST FPS
15674 101074  000310                        .WORD   310             ;RESULT FPS
15675                               ;
15676 101076  000167  000126                JMP     HOP19           ;GET OVER SUBROUTINE
15677                               ;
15678                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15679                               ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15680                               ;LDCID, LDCLD
15681                               ;
```

```
15682                                   ;               FSRC
15683                                   ;               RESULT
15684                                   ;               FPS BEFORE EXECUTION
15685                                   ;               FPS AFTER EXECUTION
15686                                   ;
15687                                   ;NO TRAP CAN OCCUR
15688                                   ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15689                                   ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15690                                   ;
15691 101102  012602                   LCDSUB: MOV     (SP)+,R2                ; RETURN ADDRESS TO USE AS POINTER
15692 101104  012737  101212  000244           MOV     #50$,@#FPVEC            ;REDIRECT TRAP VECTOR
15693 101112  012701  003142                   MOV     #RECDST,R1              ;POINT TO RESULT AREA
15694 101116  016200  000014                   MOV     14(R2),R0               ;GET TEST FPS
15695 101122  170100                           LDFPS   R0                      ;LOAD TEST FPS
15696 101124  010204                           MOV     R2,R4                   ;POINT TO TEST DATA
15697                                   ;
15698 101126  177014                   40$:    LDCID   (R4),AC0                ;*TEST INSTRUCTION (ACCORDING TO MODE)
15699                                   ;
15700                                   ;VERIFY STATUS
15701                                   ;
15702 101130  170203                   2$:     STFPS   R3                      ;SAVE FPS
15703 101132  012700  000200                   MOV     #200,R0                 ;SET FPP STATUS TO DOUBLE
15704 101136  170100                           LDFPS   R0                      ;
15705 101140  174011                           STD     AC0,(R1)                ;SAVE TEST RESULT INTO RECDST
15706 101142  016200  000016                   MOV     16(R2),R0                       ;GET EXPECTED STATUS
15707 101146  020003                           CMP     R0,R3                   ;VERIFY STATUS
15708 101150  001403                           BEQ     3$                      ;BRANCH IF GOOD
15709 101152  004737  140132                   CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15710 101156  104003                           ERROR   +3              ;FPP ERROR
15711                                                                  ;BAD FPS
15712 101160  010204                   3$:     MOV     R2,R4                   ;POINT TO EXPECTED DATA
15713 101162  062704  000004                   ADD     #4,R4
15714 101166  004767  036706           4$:     JSR     R7,DATVER                       ;VERIFY DATA
15715 101172  005767  101722                   TST     COUNT
15716 101176  001403                           BEQ     5$                      ;BRANCH IF GOOD
15717 101200  004737  140132                   CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15718 101204  104003                           ERROR   +3              ;FPP ERROR
15719                                                                  ;BAD AC0
15720 101206  000162  000020           5$:     JMP     20(R2)                  ;RETURN FROM TEST
15721                                   ;
15722                                   ;INSTRUCTION TRAPPED
15723                                   ;
15724 101212  012600                   50$:    MOV     (SP)+,R0                ;SAVE PC
15725 101214  012605                           MOV     (SP)+,R5                ;SAVE PS
15726 101216  004737  140132                   CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15727 101222  104003                           ERROR   +3              ;FPP ERROR
15728                                                                  ;INSTRUCTION WASNT SUPPOSE TO TRAP
15729 101224  000167  177756                   JMP     5$                      ;CONTINUE
15730                                   ;
15731 101230                           HOP19:
15734                                   ;
15735                                   ;--------------------------------
15736                                   ;TEST LDEXP
15737                                   ;DOUBLE
15738                                   ;
15739 101230                           MLXP:
15740                                   ;
```

FLOATING POINT TESTS

```
15741                                   ;1/EXP=10 - AC=NEG
15742 101230  005037  003030                      CLR     @#FLAG                  ;NO INTERRUPTS
15743 101234  004767  001140                      JSR     R7,LXPSUB               ;DO TEST
15744 101240  123456  067012  025252              .WORD   123456,67012,25252,171717       ;AC0
15745 101250  000010                              .WORD   10                      ;EXP
15746 101252  142056  067012  025252              .WORD   142056,67012,25252,171717               ;RESULT
15747 101262  007757                              .WORD   7757                    ; TEST FPS
15748 101264  007750                              .WORD   7750                    ;RESULT FPS
15749                                   ;2/EXP=177 - AC0=POS
15750 101266  005037  003030                      CLR     @#FLAG                  ;NO INTERRUPTS
15751 101272  004767  001102                      JSR     R7,LXPSUB               ;DO TEST
15752 101276  023456  070123  100000              .WORD   23456,70123,100000,1            ;AC0
15753 101306  000177                              .WORD   177                     ;EXP
15754 101310  077656  070123  100000              .WORD   77656,70123,100000,1            ;RESULT
15755 101320  007700                              .WORD   7700                    ; TEST FPS
15756 101322  007700                              .WORD   7700                    ;RESULT FPS
15757                                   ;3/EXP=56
15758 101324  005037  003030                      CLR     @#FLAG                  ;NO INTERRUPTS
15759 101330  004767  001044                      JSR     R7,LXPSUB               ;DO TEST
15760 101334  055555  044444  033333              .WORD   55555,44444,33333,22222         ;AC0
15761 101344  000056                              .WORD   56                      ;EXP
15762 101346  053555  044444  033333              .WORD   53555,44444,33333,22222 ;RESULT
15763 101356  007757                              .WORD   7757                    ; TEST FPS
15764 101360  007740                              .WORD   7740                    ;RESULT FPS
15765                                   ;4/EXP=-151, AC0=UV
15766 101362  005037  003030                      CLR     @#FLAG                  ;NO INTERRUPTS
15767 101366  004767  001006                      JSR     R7,LXPSUB               ;DO TEST
15768 101372  100077  177777  177777              .WORD   100077,-1,-1,-2         ;AC0
15769 101402  177623                              .WORD   -155                    ;EXP
15770 101404  104677  177777  177777              .WORD   104677,-1,-1,-2         ;RESULT
15771 101414  007757                              .WORD   7757                    ; TEST FPS
15772 101416  007750                              .WORD   7750                    ;RESULT FPS
15773                                   ;5/EXP=-177
15774 101420  005037  003030                      CLR     @#FLAG                  ;NO INTERRUPTS
15775 101424  004767  000750                      JSR     R7,LXPSUB               ;DO TEST
15776 101430  000177  177777  177777              .WORD   177,-1,-1,-2            ;AC0
15777 101440  177601                              .WORD   -177                    ;EXP
15778 101442  000377  177777  177777              .WORD   377,-1,-1,-2            ;RESULT
15779 101452  007700                              .WORD   7700                    ; TEST FPS
15780 101454  007700                              .WORD   7700                    ;RESULT FPS
15781                                   ;6/EXP=-200, UNDERFLOW
15782 101456  012737  000001  003030              MOV     #1,@#FLAG               ; INTERRUPTS
15783 101464  004767  000710                      JSR     R7,LXPSUB               ;DO TEST
15784 101470  030131  032334  035363              .WORD   30131,32334,35363,73031         ;AC0
15785 101500  177600                              .WORD   -200                    ;EXP
15786 101502  000131  032334  035363              .WORD   131,32334,35363,73031           ;RESULT
15787 101512  007740                              .WORD   7740                    ; TEST FPS
15788 101514  107744                              .WORD   107744                  ;RESULT FPS
15789 101516  000012                              .WORD   12                      ;FEC
15790                                   ;7/EXP=LARGEST NEGATIVE
15791 101520  012737  000001  003030              MOV     #1,@#FLAG               ;EXPECT  INTERRUPTS
15792 101526  004767  000646                      JSR     R7,LXPSUB               ;DO TEST
15793 101532  000000  000123  000456              .WORD   0,123,456,1             ;AC0
15794 101542  100000                              .WORD   100000                  ;EXP
15795 101544  040000  000123  000456              .WORD   40000,123,456,1         ;RESULT
15796 101554  002200                              .WORD   2200                    ; TEST FPS
15797 101556  102200                              .WORD   102200                  ;RESULT FPS
```

FLOATING POINT TESTS

```
15798 101560  000012                        .WORD   12              ;FEC
15799                             ;8/EXP=-200, NEG. AC0
15800 101562  012737  000001  003030        MOV     #1,@#FLAG               ; INTERRUPTS
15801 101570  004767  000604                JSR     R7,LXPSUB               ;DO TEST
15802 101574  111111  100000  100000        .WORD   111111,100000,100000,-1         ;AC0
15803 101604  177600                        .WORD   -200            ;EXP
15804 101606  100111  100000  100000        .WORD   100111,100000,100000,-1         ;RESULT
15805 101616  002217                        .WORD   2217            ; TEST FPS
15806 101620  102214                        .WORD   102214          ;RESULT FPS
15807 101622  000012                        .WORD   12              ;FEC
15808                             ;9/EXP=-1743, FIU=0
15809 101624  012737  000002  003030        MOV     #2,@#FLAG               ;NO INTERRUPTS
15810 101632  004767  000542                JSR     R7,LXPSUB               ;DO TEST
15811 101636  123456  012346  012346        .WORD   123456,12346,12346,123          ;AC0
15812 101646  176035                        .WORD   -1743           ;EXP
15813 101650  000000  000000  000000        .WORD   0,0,0,0         ;RESULT
15814 101660  005700                        .WORD   5700            ; TEST FPS
15815 101662  005704                        .WORD   5704            ;RESULT FPS
15816 101664  000012                        .WORD   12              ;FEC
15817                             ;10/EXP =-16616, FID=1
15818 101666  012737  000002  003030        MOV     #2,@#FLAG               ;NO INTERRUPTS
15819 101674  004767  000500                JSR     R7,LXPSUB               ;DO TEST
15820 101700  000377  123456  065432        .WORD   377,123456,65432,1              ;AC0
15821 101710  161162                        .WORD   -16616          ;EXP
15822 101712  074577  123456  065432        .WORD   74577,123456,65432,1            ;RESULT
15823 101722  047700                        .WORD   47700           ; TEST FPS
15824 101724  147700                        .WORD   147700          ;RESULT FPS
15825 101726  000012                        .WORD   12              ;FEC
15826                           ;11/EXP=177, AC0=UNDEFINED VARIABLE
15827 101730  005037  003030                CLR     @#FLAG          ;NO INTERRUPTS
15828 101734  004767  000440                JSR     R7,LXPSUB               ;DO TEST
15829 101740  100177  177777  177777        .WORD   100177,-1,-1,-1         ;AC0
15830 101750  000177                        .WORD   177             ;EXP
15831 101752  177777  177777  177777        .WORD   -1,-1,-1,-1             ;RESULT
15832 101762  007700                        .WORD   7700            ; TEST FPS
15833 101764  007710                        .WORD   7710            ;RESULT FPS
15834                           ;12/EXP=150 AC0=POS
15835 101766  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
15836 101772  004767  000402                JSR     R7,LXPSUB               ;DO TEST
15837 101776  000200  000100  000200        .WORD   200,100,200,300         ;AC0
15838 102006  000150                        .WORD   150             ;EXP
15839 102010  072000  000100  000200        .WORD   72000,100,200,300                       ;RESULT
15840 102020  007717                        .WORD   7717            ; TEST FPS
15841 102022  007700                        .WORD   7700            ;RESULT FPS
15842                           ;13/EXP=200, AC0=NEG
15843 102024  012737  000001  003030        MOV     #1,@#FLAG
15844 102032  004767  000342                JSR     R7,LXPSUB               ;DO TEST
15845 102036  177777  177777  177777        .WORD   -1,-1,-1,-1             ;AC0
15846 102046  000200                        .WORD   200             ;EXP
15847 102050  100177  177777  177777        .WORD   100177,-1,-1,-1         ;RESULT
15848 102060  007705                        .WORD   7705            ; TEST FPS
15849 102062  107716                        .WORD   107716          ;RESULT FPS
15850 102064  000010                        .WORD   10              ;FEC
15851                             ;14/EXP=400, FID
15852 102066  012737  000002  003030        MOV     #2,@#FLAG               ;INTERRUPT
15853 102074  004767  000300                JSR     R7,LXPSUB               ;DO TEST
15854 102100  000555  177777  177776        .WORD   555,-1,-2,-3            ;AC0
```

FLOATING POINT TESTS

```
15855 102110  000400                        .WORD   400                     ;EXP
15856 102112  040155  177777  177776        .WORD   40155,-1,-2,-3          ;RESULT
15857 102122  047700                        .WORD   47700           ; TEST FPS
15858 102124  147702                        .WORD   147702          ;RESULT FPS
15859 102126  000010                        .WORD   10              ;FEC
15860                       ;15/EXP=11011 FIU=0
15861 102130  012737  000000  003030        MOV     #0,@#FLAG               ;NO INTERRUPT
15862 102136  004767  000236                JSR     R7,LXPSUB               ;DO TEST
15863 102142  177773  177777  177776        .WORD   177773,-1,-2,-3         ;ACO
15864 102152  011011                        .WORD   11011           ;EXP
15865 102154  000000  000000  000000        .WORD   0,0,0,0         ;RESULT
15866 102164  006700                        .WORD   6700            ; TEST FPS
15867 102166  006706                        .WORD   6706            ;RESULT FPS
15868                       ;16/EXP=LARGEST POSITIVE
15869 102170  012737  000001  003030        MOV     #1,@#FLAG               ;INTERRUPT
15870 102176  004767  000176                JSR     R7,LXPSUB               ;DO TEST
15871 102202  123456  000100  000100        .WORD   123456,100,100,200              ;ACO
15872 102212  077777                        .WORD   77777           ;EXP
15873 102214  137656  000100  000100        .WORD   137656,100,100,200              ;RESULT
15874 102224  007740                        .WORD   7740            ; TEST FPS
15875 102226  107752                        .WORD   107752          ;RESULT FPS
15876 102230  000010                        .WORD   10              ;FEC
15877                       ;17/FLOATING
15878 102232  005037  003030                CLR     @#FLAG          ;NO INTERRUPT
15879 102236  004767  000136                JSR     R7,LXPSUB               ;DO TEST
15880 102242  123456  023465  000555        .WORD   123456,23465,555,444            ;ACO
15881 102252  000050                        .WORD   50              ;EXP
15882 102254  152056  023465  000555        .WORD   152056,23465,555,444            ;RESULT
15883 102264  007500                        .WORD   7500            ; TEST FPS
15884 102266  007510                        .WORD   7510            ;RESULT FPS
15885                       ;18/FLOATING UNDERFLOW
15886 102270  012737  000001  003030        MOV     #1,@#FLAG               ;INTERRUPT
15887 102276  004767  000076                JSR     R7,LXPSUB               ;DO TEST
15888 102302  000333  000444  000555        .WORD   333,444,555,666         ;ACO
15889 102312  177600                        .WORD   -200            ;EXP
15890 102314  000133  000444  000555        .WORD   133,444,555,666         ;RESULT
15891 102324  007500                        .WORD   7500            ; TEST FPS
15892 102326  107504                        .WORD   107504          ;RESULT FPS
15893 102330  000012                        .WORD   12              ;FEC
15894                       ;19/FLOATING OVERFLOW
15895 102332  012737  000001  003030        MOV     #1,@#FLAG               ;INTERRUPT
15896 102340  004767  000034                JSR     R7,LXPSUB               ;DO TEST
15897 102344  012346  000123  000345        .WORD   12346,123,345,456               ;ACO
15898 102354  000400                        .WORD   400             ;EXP
15899 102356  040146  000123  000345        .WORD   40146,123,345,456               ;RESULT
15900 102366  007400                        .WORD   7400            ; TEST FPS
15901 102370  107402                        .WORD   107402          ;RESULT FPS
15902 102372  000010                        .WORD   10              ;FEC
15903                       ;
15904 102374  000167  000250                JMP     HOP20                   ;GET OVER SUBROUTINE
15905                       ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15906                       ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15907                       ;LDEXP
15908                       ;               ACO
15909                       ;               EXPONENT
15910                       ;               RESULT
15911                       ;               FPS BEFORE EXECUTION
```

```
15912                                   ;               FPS AFTER EXECUTION
15913                                   ;               (FEC)
15914                                   ;
15915                                   ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15916                                   ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
15917                                   ;
15918 102400  012602                   LXPSUB: MOV     (SP)+,R2                ; RETURN ADDRESS TO USE AS POINTER
15919 102402  012737  102470  000244           MOV     #50$,@#FPVEC            ;REDIRECT TRAP VECTOR
15920 102410  012701  003142                   MOV     #RECDST,R1              ;POINT TO RESULT AREA
15921 102414  012700  000200                   MOV     #200,R0                 ;SET FPS TO DOUBLE
15922 102420  170100                           LDFPS   R0                      ;
15923 102422  010204                           MOV     R2,R4                   ;POINT TO ACO DATA
15924 102424  172414                           LDD     (R4),AC0                        ;LOAD ACO
15925 102426  016200  000022                   MOV     22(R2),R0               ;GET TEST FPS
15926 102432  170100                           LDFPS   R0                      ;LOAD TEST FPS
15927 102434  016204  000010                   MOV     10(R2),R4               ;POINT TO TEST DATA
15928                                   ;
15929 102440  176404                   40$:    LDEXP   R4,AC0                  ;*TEST INSTRUCTION (ACCORDING TO MODE)
15930 102442  170327                   1$:     STST    (PC)+                   ;WAIT FOR POSSIBLE FPA TRAP.
15931 102444  000000                           .WORD   0                       ;STORE STATUS HERE
15932                                   ;
15933                                   ;INSTRUCTION DIDNT TRAP
15934                                   ;
15935 102446  032737  000001  003030           BIT     #1,@#FLAG                       ;VERIFY A NO TRAP CONDITION
15936 102454  001426                           BEQ     2$                      ;BRANCH IF GOOD
15937 102456  004737  140132                   CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15938 102462  104003                           ERROR   +3                      ;FPP ERROR
15939                                                                           ;INSTRUCTION SHOULD HAVE TRAPPED
15940 102464  000167  000042                   JMP     2$                              ;REJOIN CODE
15941                                   ;
15942                                   ;INSTRUCTION TRAPPED
15943                                   ;
15944 102470  032737  000001  003030   50$:    BIT     #1,@#FLAG               ;SEE IF EXPECTING A TRAP
15945 102476  001005                           BNE     51$                     ;BRANCH IF EXPECTING A TRAP
15946 102500  004737  140132                   CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15947 102504  104003                           ERROR   +3                      ;FPP ERROR
15948                                                                           ;INSTRUCTION WASNT SUPPOSE TO TRAP
15949 102506  000167  000020                   JMP     2$                              ;REJOIN CODE
15950 102512  012604                   51$:    MOV     (SP)+,R4                ;SEE IF PC = INSTRUCTION
15951 102514  005726                           TST     (SP)+                   ;CLEAN UP STACK
15952 102516  022704  102442                   CMP     #1$,R4                  ;
15953 102522  001403                           BEQ     2$                      ;BRANCH IF GOOD COMPARE
15954 102524  004737  140132                   CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15955 102530  104003                           ERROR   +3                      ;FPP ERROR
15956                                                                           ;PC WAS INCORRECT
15957                                   ;
15958                                   ;COMMON CODE FOR TRAP AND NO TRAP
15959                                   ;VERIFY STATUS
15960                                   ;
15961 102532  170203                   2$:     STFPS   R3                      ;SAVE FPS
15962 102534  012700  000200                   MOV     #200,R0                 ;SETUP FPS
15963 102540  170100                           LDFPS   R0                      ;FPS=200
15964 102542  174011                           STD     AC0,(R1)                ;GET RESULT
15965 102544  016200  000024                   MOV     24(R2),R0                       ;GET EXPECTED STATUS
15966 102550  020003                           CMP     R0,R3                   ;VERIFY STATUS
15967 102552  001403                           BEQ     3$                      ;BRANCH IF GOOD
15968 102554  004737  140132                   CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
```

FLOATING POINT TESTS

```
15969 102560  104003                          ERROR   +3              ;FPP ERROR
15970                                                                 ;BAD FPS
15971 102562  010204                    3$:   MOV     R2,R4                   ;POINT TO EXPECTED DATA
15972 102564  062704  000012                    ADD     #12,R4
15973 102570  004767  035304              4$:   JSR     R7,DATVER                       ;VERIFY DATA
15974 102574  005767  100320                    TST     COUNT
15975 102600  001403                          BEQ     5$                      ;BRANCH IF GOOD
15976 102602  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15977 102606  104003                          ERROR   +3              ;FPP ERROR
15978                                                                 ;BAD ACO
15979 102610  005737  003030              5$:   TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
15980 102614  001002                          BNE     7$                      ;BRANCH IF NEED TO CHECK
15981 102616  000162  000026                    JMP     26(R2)                  ;RETURN FROM TEST
15982                                     ;VERIFY FEC
15983 102622  012704  003122              7$:   MOV     #RECFEC,R4              ;POINT TO FEC AREA
15984 102626  170314                          STST    (R4)                    ;SAVE FEC
15985 102630  021462  000026                    CMP     (R4),26(R2)                     ;VERIFY FEC FOR OVERFLOW
15986 102634  001403                          BEQ     8$                      ;BRANCH IF GOOD
15987 102636  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
15988 102642  104003                          ERROR   +3              ;FPP ERROR
15989                                                                 ;BAD FEC
15990 102644  000162  000030              8$:   JMP     30(R2)          ;RETURN FROM TEST
15991                                     ;
15992 102650                              HOP20:
15995                                     ;
15996                                     ;---------------------------------
15997                                     ;TEST STCDI, STCDL
15998                                     ;
15999 102650                              MSCD:
16000                                     ;
16001                                     ;1/ACO=0, INT
16002 102650  005037  003030                    CLR     @#FLAG          ;NO INTERRUPTS
16003 102654  004767  000610                    JSR     R7,SCDSUB               ;DO TEST
16004 102660  000177  000000  000000            .WORD   0177,0,0,0              ;ACO
16005 102670  000000  177777                    .WORD   0,-1            ;RESULT
16006 102674  007640                            .WORD   7640            ; TEST FPS
16007 102676  007644                            .WORD   7644            ;RESULT FPS
16008                                     ;2/ACO=-0, LONG
16009 102700  005037  003030                    CLR     @#FLAG          ;INTERRUPT
16010 102704  004767  000560                    JSR     R7,SCDSUB               ;DO TEST
16011 102710  100177  177777  177777            .WORD   100177,-1,-1,-1         ;ACO
16012 102720  000000  000000                    .WORD   0,0             ;RESULT
16013 102724  007700                            .WORD   7700            ; TEST FPS
16014 102726  007704                            .WORD   7704            ;RESULT FPS
16015                                     ;3/EXP=100, LONG
16016 102730  005037  003030                    CLR     @#FLAG                  ;NO INTERRUPT
16017 102734  004767  000530                    JSR     R7,SCDSUB               ;DO TEST
16018 102740  020000  000000  000000            .WORD   20000,0,0,0             ;ACO
16019 102750  000000  000000                    .WORD   0,0             ;RESULT
16020 102754  000300                            .WORD   300             ; TEST FPS
16021 102756  000304                            .WORD   304             ;RESULT FPS
16022                                     ;4/EXP=200, BAISED 0, INT, ROUND
16023 102760  005037  003030                    CLR     @#FLAG          ;INTERRUPT
16024 102764  004767  000500                    JSR     R7,SCDSUB               ;DO TEST
16025 102770  140177  177777  000001            .WORD   140177,177777,1,1                       ;ACO
16026 103000  000000  000000                    .WORD   0,0             ;RESULT
16027 103004  007700                            .WORD   7700            ; TEST FPS
```

FLOATING POINT TESTS

```
16028 103006  007704                        .WORD   7704            ;RESULT FPS
16029                              ;5/LONG
16030 103010  005037  003030                 CLR     @#FLAG          ;INTERRUPT
16031 103014  004767  000450                 JSR     R7,SCDSUB               ;DO TEST
16032 103020  047667  075757  157737         .WORD   47667,75757,157737,167773               ;ACO
16033 103030  055675  173757                 .WORD   55675,173757            ;RESULT
16034 103034  007717                        .WORD   7717            ; TEST FPS
16035 103036  007700                        .WORD   7700            ;RESULT FPS
16036                              ;6/LONG, EXP=2**32
16037 103040  005037  003030                 CLR     @#FLAG                  ;NO INTERRUPT
16038 103044  004767  000420                 JSR     R7,SCDSUB               ;DO TEST
16039 103050  046400  000000  000000         .WORD   46400,0,0,0             ;ACO
16040 103060  001000  000000                 .WORD   1000,0          ;RESULT
16041 103064  007700                        .WORD   7700            ; TEST FPS
16042 103066  007700                        .WORD   7700            ;RESULT FPS
16043                              ;7/LONG, EXP>2**32
16044 103070  012737  000001  003030         MOV     #1,@#FLAG               ;INTERRUPT
16045 103076  004767  000366                 JSR     R7,SCDSUB               ;DO TEST
16046 103102  077607  000000  000000         .WORD   77607,0,0,0             ;ACO
16047 103112  000000  000000                 .WORD   0,0             ;RESULT
16048 103116  007700                        .WORD   7700            ; TEST FPS
16049 103120  107705                        .WORD   107705          ;RESULT FPS
16050                              ;8/INT, EXP=2**15
16051 103122  005037  003030                 CLR     @#FLAG          ;NO INTERRUPTS
16052 103126  004767  000336                 JSR     R7,SCDSUB               ;DO TEST
16053 103132  043200  000000  000000         .WORD   43200,0,0,0             ;ACO
16054 103142  010000  177777                 .WORD   10000,-1                ;RESULT
16055 103146  007600                        .WORD   7600            ; TEST FPS
16056 103150  007600                        .WORD   7600            ;RESULT FPS
16057                              ;9/INT, EXP>2**15
16058 103152  012737  000001  003030         MOV     #1,@#FLAG               ;INTERRUPT
16059 103160  004767  000304                 JSR     R7,SCDSUB               ;DO TEST
16060 103164  077777  177777  177777         .WORD   77777,-1,-1,-1          ;ACO
16061 103174  000000  177777                 .WORD   0,-1            ;RESULT
16062 103200  007600                        .WORD   7600            ; TEST FPS
16063 103202  107605                        .WORD   107605          ;RESULT FPS
16064                              ;10/INT, EXP>2**15, FID
16065 103204  012737  000000  003030         MOV     #0,@#FLAG               ;NO INTERRUPT
16066 103212  004767  000252                 JSR     R7,SCDSUB               ;DO TEST
16067 103216  043300  000000  000000         .WORD   43300,0,0,0             ;ACO
16068 103226  000000  014000                 .WORD   0,14000         ;RESULT
16069 103232  047700                        .WORD   47700           ; TEST FPS
16070 103234  047700                        .WORD   47700           ;RESULT FPS
16071                              ;11/INT, EXP>2**15, FIC=0
16072 103236  012737  000000  003030         MOV     #0,@#FLAG               ;NO INTERRUPT
16073 103244  004767  000220                 JSR     R7,SCDSUB               ;DO TEST
16074 103250  143300  177777  177777         .WORD   143300,-1,-1,-1         ;ACO
16075 103260  177777  163741                 .WORD   -1,163741               ;RESULT
16076 103264  007300                        .WORD   7300            ; TEST FPS
16077 103266  007310                        .WORD   7310            ;RESULT FPS
16078                              ;12/LONG, EXP>2**32, FID
16079 103270  012737  000002  003030         MOV     #2,@#FLAG               ;INTERRUPT
16080 103276  004767  000166                 JSR     R7,SCDSUB               ;DO TEST
16081 103302  050100  000000  000000         .WORD   50100,0,0,0             ;ACO
16082 103312  000000  000000                 .WORD   0,0             ;RESULT
16083 103316  047700                        .WORD   47700           ; TEST FPS
16084 103320  147705                        .WORD   147705          ;RESULT FPS
```

```
16085
16086 103322  012737  000000  003030          ;13/LONG, EXP>2**32, FIC=0
16087 103330  004767  000134                  MOV     #0,@#FLAG               ;NO INTERRUPT
16088 103334  050377  177777  177777          JSR     R7,SCDSUB               ;DO TEST
16089 103344  000000  000000                  .WORD   50377,-1,-1,-1          ;ACO
16090 103350  007300                          .WORD   0,0             ;RESULT
16091 103352  007305                          .WORD   7300            ; TEST FPS
16092                                         .WORD   7305            ;RESULT FPS
16093 103354  005037  003030          ;14/LONG, EXP<0
16094 103360  004767  000104                  CLR     @#FLAG          ;NO INTERRUPTS
16095 103364  100200  177777  177777          JSR     R7,SCDSUB               ;DO TEST
16096 103374  000000  000000                  .WORD   100200,-1,-1,-1         ;ACO
16097 103400  007757                          .WORD   0,0             ;RESULT
16098 103402  007744                          .WORD   7757            ; TEST FPS
16099                                         .WORD   7744            ;RESULT FPS
16100 103404  005037  003030          ;15/INT, EXP<0
16101 103410  004767  000054                  CLR     @#FLAG          ;NO INTERRUPTS
16102 103414  037700  177777  177777          JSR     R7,SCDSUB               ;DO TEST
16103 103424  000000  177777                  .WORD   37700,-1,-1,-2          ;ACO
16104 103430  007600                          .WORD   0,-1            ;RESULT
16105 103432  007604                          .WORD   7600            ; TEST FPS
16106                                         .WORD   7604            ;RESULT FPS
16107 103434  005037  003030          ;16/INT, EXP=10
16108 103440  004767  000024                  CLR     @#FLAG          ;NO INTERRUPTS
16109 103444  004377  177777  177777          JSR     R7,SCDSUB               ;DO TEST
16110 103454  000000  177777                  .WORD   4377,-1,-1,-1           ;ACO
16111 103460  007600                          .WORD   0,-1            ;RESULT
16112 103462  007604                          .WORD   7600            ; TEST FPS
16113                                         .WORD   7604            ;RESULT FPS
16114 103464  000167  000244          ;
                                              JMP     HOP21           ;GET OVER SUBROUTINE
16115                                 ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
16116                                 ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
16117                                 ;STCDI, STCDL, STCFI, STCFL
16118                                 ;
16119                                 ;               ACO
16120                                 ;               RESULT
16121                                 ;               FPS BEFORE EXECUTION
16122                                 ;               FPS AFTER EXECUTION
16123                                 ;               (FEC)
16124                                 ;
16125                                 ;TRAP ON CONVERSION FAILURE
16126                                 ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
16127                                 ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
16128                                 ;
16129 103470  012602                  SCDSUB: MOV     (SP)+,R2                ; RETURN ADDRESS TO USE AS POINTER
16130 103472  012737  103564  000244          MOV     #50$,@#FPVEC            ;REDIRECT TRAP VECTOR
16131 103500  012701  003144                  MOV     #RECDST+2,R1            ;POINT TO RESULT AREA
16132 103504  012711  177777                  MOV     #-1,(R1)                ;PRELOAD RECEIVE DATA BUFFER
16133 103510  012741  177777                  MOV     #-1,-(R1)               ;
16134 103514  012700  000200                  MOV     #200,R0                 ;SET FPS TO DOUBLE
16135 103520  170100                          LDFPS   R0                      ;
16136 103522  010204                          MOV     R2,R4                   ;POINT TO ACO DATA
16137 103524  172414                          LDD     (R4),AC0                        ;LOAD ACO
16138 103526  016200  000014                  MOV     14(R2),R0               ;GET TEST FPS
16139 103532  170100                          LDFPS   R0                      ;LOAD TEST FPS
16140                                         ;
16141 103534  175411                  40$:    STCDI   AC0,(R1)                ;*TEST INSTRUCTION(ACCORDING TO MODE)
```

FLOATING POINT TESTS

```
16142 103536  170327                1$:     STST    (PC)+                   ;WAIT FOR POSSIBLE FPA TRAP.
16143 103540  000000                        .WORD   0                       ;STORE STATUS HERE.
16144                               ;
16145                               ;INSTRUCTION DIDNT TRAP
16146                               ;
16147 103542  032737  000001 003030         BIT     #1,@#FLAG                       ;VERIFY A NO TRAP CONDITION
16148 103550  001426                        BEQ     2$                      ;BRANCH IF GOOD
16149 103552  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
16150 103556  104003                        ERROR   .3                      ;FPP ERROR
16151                                                                       ;INSTRUCTION SHOULD HAVE TRAPPED
16152 103560  000167  000042                JMP     2$                      ;REJOIN CODE
16153                               ;
16154                               ;INSTRUCTION TRAPPED
16155                               ;
16156 103564  032737  000001 003030 50$:    BIT     #1,@#FLAG               ;SEE IF EXPECTING A TRAP
16157 103572  001005                        BNE     51$                     ;BRANCH IF EXPECTING A TRAP
16158 103574  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
16159 103600  104003                        ERROR   .3                      ;FPP ERROR
16160                                                                       ;INSTRUCTION WASNT SUPPOSE TO TRAP
16161 103602  000167  000020                JMP     2$                      ;REJOIN CODE
16162 103606  012604                51$:    MOV     (SP)+,R4                ;SEE IF PC = INSTRUCTION
16163 103610  005726                        TST     (SP)+                   ;CLEAN UP STACK
16164 103612  022704  103536                CMP     #1$,R4                  ;
16165 103616  001403                        BEQ     2$                      ;BRANCH IF GOOD COMPARE
16166 103620  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
16167 103624  104003                        ERROR   .3                      ;FPP ERROR
16168                                                                       ;PC WAS INCORRECT
16169                               ;
16170                               ;COMMON CODE FOR TRAP AND NO TRAP
16171                               ;VERIFY STATUS
16172                               ;
16173 103626  170203                2$:     STFPS   R3                      ;SAVE FPS
16174 103630  016200  000016                MOV     16(R2),R0                       ;GET EXPECTED STATUS
16175 103634  020003                        CMP     R0,R3                   ;VERIFY STATUS
16176 103636  001403                        BEQ     3$                      ;BRANCH IF GOOD
16177 103640  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
16178 103644  104003                        ERROR   .3                      ;FPP ERROR
16179                                                                       ;BAD FPS
16180 103646  010204                3$:     MOV     R2,R4                   ;POINT TO EXPECTED DATA
16181 103650  062704  000010                ADD     #10,R4
16182 103654  004767  034202          4$:   JSR     R7,DATVFR                       ;VERIFY DATA
16183 103660  005767  077234                TST     COUNT
16184 103664  001403                        BEQ     5$                      ;BRANCH IF GOOD
16185 103666  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
16186 103672  104003                        ERROR   .3                      ;FPP ERROR
16187                                                                       ;BAD ACO
16188 103674  005737  003030          5$:   TST     @#FLAG                  ;SEE IF NEED TO CHECK FEC
16189 103700  001002                        BNE     7$                      ;BRANCH IF NEED TO CHECK
16190 103702  000162  000020                JMP     20(R2)                  ;RETURN FROM TEST
16191                               ;VERIFY FEC
16192 103706  012704  003122          7$:   MOV     #RECFEC,R4              ;POINT TO FEC AREA
16193 103712  170314                        STST    (R4)                    ;SAVE FEC
16194 103714  021427  000006                CMP     (R4),#6                 ;VERIFY FEC FOR OVERFLOW
16195 103720  001403                        BEQ     8$                      ;BRANCH IF GOOD
16196 103722  004737  140132                CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
16197 103726  104003                        ERROR   .3                      ;FPP ERROR
16198                                                                       ;BAD FEC
```

FLOATING POINT TESTS

```
16199 103730  000162  000020          8$:     JMP     20(R2)          ;RETURN FROM TEST
16200                                 ;
16201 103734                          HOP21:
16202                                 ;
16205                                 ;
16206                                 ;---------------------------------
16207                                 ;TEST STCFI, STCFL
16208                                 ;
16209 103734                          MSCF:
16210                                 ;
16211                                 ;1/LONG EXP =30
16212 103734  005037  003030                  CLR     @#FLAG          ;NO INTERRUPTS
16213 103740  004767  177524                  JSR     R7,SCDSUB               ;DO TEST
16214 103744  044541  052525  177777          .WORD   44541,52525,-1,-1               ;AC0
16215 103754  000003  102525                  .WORD   3,102525                ;RESULT
16216 103760  007517                          .WORD   7517            ; TEST FPS
16217 103762  007500                          .WORD   7500            ;RESULT FPS
16218                                 ;2/INT, EXP<0
16219 103764  005037  003030                  CLR     @#FLAG          ;NO INTERRUPTS
16220 103770  004767  177474                  JSR     R7,SCDSUB               ;DO TEST
16221 103774  002300  177777  177777          .WORD   2300,-1,-1,-1           ;AC0
16222 104004  000000  177777                  .WORD   0,-1            ;RESULT
16223 104010  007400                          .WORD   7400            ; TEST FPS
16224 104012  007404                          .WORD   7404            ;RESULT FPS
16225                                 ;3/LONG, EXP
16226 104014  012737  000001  003030          MOV     #1,@#FLAG               ;INTERRUPT
16227 104022  004767  177442                  JSR     R7,SCDSUB               ;DO TEST
16228 104026  070000  177777  177777          .WORD   70000,-1,-1,-1          ;AC0
16229 104036  000000  000000                  .WORD   0,0             ;RESULT
16230 104042  007540                          .WORD   7540            ; TEST FPS
16231 104044  107545                          .WORD   107545          ;RESULT FPS
16232                                 ;4/INT,EXP=5, FIC=0, FID=1
16233 104046  005037  003030                  CLR     @#FLAG          ;NO INTERRUPTS
16234 104052  004767  177412                  JSR     R7,SCDSUB               ;DO TEST
16235 104056  052000  000000  177777          .WORD   52000,0,-1,-1           ;AC0
16236 104066  000000  177777                  .WORD   0,-1            ;RESULT
16237 104072  047000                          .WORD   47000           ; TEST FPS
16238 104074  047005                          .WORD   47005           ;RESULT FPS
16239                                 ;
16242                                 ;
16243                                 ;---------------------------------
16244                                 ;TEST STEXP
16245                                 ;
16246 104076                          MSXP:
16247                                 ;
16248                                 ;1/EXP=100
16249 104076  004767  000154                  JSR     R7,SXPSUB               ;DO TEST
16250 104102  020000  000000  000000          .WORD   20000,0,0,0             ;AC0
16251 104112  177700                          .WORD   -100            ;RESULT
16252 104114  007740                          .WORD   7740            ; TEST FPS
16253 104116  007750                          .WORD   7750            ;RESULT FPS
16254                                 ;2/EXP=201 FLOAT, NEG
16255 104120  004767  000132                  JSR     R7,SXPSUB               ;DO TEST
16256 104124  140377  177777  177777          .WORD   140377,-1,-1,0          ;AC0
16257 104134  000001                          .WORD   1               ;RESULT
16258 104136  007500                          .WORD   7500            ; TEST FPS
16259 104140  007500                          .WORD   7500            ;RESULT FPS
```

FLOATING POINT TESTS

```
16260                                   ;3/EXP=-177
16261 104142  004767  000110                      JSR     R7,SXPSUB               ;DO TEST
16262 104146  000177  177777  177777              .WORD   177,-1,-1,-1            ;ACO
16263 104156  177600                              .WORD   177600          ;RESULT
16264 104160  007700                              .WORD   7700            ; TEST FPS
16265 104162  007710                              .WORD   7710            ;RESULT FPS
16266                                   ;4/EXP=-100
16267 104164  004767  000066                      JSR     R7,SXPSUB               ;DO TEST
16268 104170  020000  000000  177777              .WORD   20000,0,-1,-1           ;ACO
16269 104200  177700                              .WORD   -100            ;RESULT
16270 104202  040200                              .WORD   40200           ; TEST FPS
16271 104204  040210                              .WORD   40210           ;RESULT FPS
16272                                   ;5/EXP=200
16273 104206  004767  000044                      JSR     R7,SXPSUB               ;DO TEST
16274 104212  040000  000000  000000              .WORD   40000,0,0,0             ;ACO
16275 104222  000000                              .WORD   0               ;RESULT
16276 104224  007700                              .WORD   7700            ; TEST FPS
16277 104226  007704                              .WORD   7704    ;RESULT FPS
16278                                   ;6/EXP=0
16279 104230  004767  000022                      JSR     R7,SXPSUB               ;DO TEST
16280 104234  000177  177777  177777              .WORD   177,-1,-1,-1            ;ACO
16281 104244  177600                              .WORD   177600          ;RESULT
16282 104246  000000                              .WORD   0               ; TEST FPS
16283 104250  000010                              .WORD   10              ;RESULT FPS
16284                                   ;
16285 104252  000167  000120                      JMP     HOP22           ;GET OVER SUBROUTINE
16286                                   ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
16287                                   ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
16288                                   ;STEXP
16289                                   ;               ACO
16290                                   ;               EXPONENT RESULT
16291                                   ;               FPS BEFORE EXECUTION
16292                                   ;               FPS AFTER EXECUTION
16293                                   ;
16294                                   ;NO TRAPS CAN OCCUR
16295                                   ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
16296                                   ;*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X*X
16297                                   ;
16298 104256  012602                    SXPSUB: MOV     (SP)+,R2                ; RETURN ADDRESS TO USE AS POINTER
16299 104260  012737  104360  000244            MOV     #50$,@#FPVEC            ;REDIRECT TRAP VECTOR
16300 104266  012701  003142                    MOV     #RECDST,R1              ;POINT TO RESULT AREA
16301 104272  012700  000200                    MOV     #200,R0                 ;SET FPS TO DOUBLE
16302 104276  170100                            LDFPS   R0                      ;
16303 104300  010204                            MOV     R2,R4                   ;POINT TO ACO DATA
16304 104302  172414                            LDD     (R4),AC0                        ;LOAD ACO
16305 104304  016200  000012                    MOV     12(R2),R0               ;GET TEST FPS
16306 104310  170100                            LDFPS   R0                      ;LOAD TEST FPS
16307                                   ;
16308 104312  175011                    40$:    STEXP   AC0,(R1)                ;*TEST INSTRUCTION(ACCORDING TO MODE)
16309                                   ;
16310                                   ;VERIFY STATUS
16311                                   ;
16312 104314  170203                    2$:     STFPS   R3                      ;SAVE FPS
16313 104316  016200  000014                    MOV     14(R2),R0                       ;GET EXPECTED STATUS
16314 104322  020003                            CMP     R0,R3                   ;VERIFY STATUS
16315 104324  001403                            BEQ     3$                      ;BRANCH IF GOOD
16316 104326  004737  140132                    CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
```

FLOATING POINT TESTS

```
16317 104332  104003                          ERROR   +3              ;FPP ERROR
16318                                                                 ;BAD FPS
16319 104334  016204  000010          3$:     MOV     10(R2),R4               ;POINT TO EXPECTED EXPONENT
16320 104340  020437  003142                  CMP     R4,@#RECDST             ;VERIFY EXPONENT
16321 104344  001403                          BEQ     5$                      ;BRANCH IF GOOD
16322 104346  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
16323 104352  104003                          ERROR   +3              ;FPP ERROR
16324                                                                 ;BAD ACO
16325 104354  000162  000016          5$:     JMP     16(R2)                  ;RETURN FROM TEST
16326                                 ;
16327                                 ;INSTRUCTION TRAPPED
16328                                 ;
16329 104360  012600                  50$:    MOV     (SP)+,R0                ;SAVE PC
16330 104362  012605                          MOV     (SP)+,R5                ;SAVE OLD PS
16331 104364  004737  140132                  CALL    @#DETFPA                ;DETERMINE FLOATING POINT FAULT. $$$
16332 104370  104003                          ERROR   +3              ;FPP ERROR
16333                                                                 ;WILD TRAP DURING STEXP
16334 104372  000167  177756                  JMP     5$                      ;REJOIN CODE
16335                                 ;
16336 104376                          HOP22:
16339
16340                                 .ENABL AMA
16341                                 .SBTTL  TEST  - CHECK REGISTER ACCESS
16342                                 ;CHECK REGISTER ACCESS - THIS TEST WILL VERIFY EACH OF THE THREE
16343                                 ;CACHE MEMORY SYSTEM REGISTERS CAN BE ACCESSED WITHOUT A TRAP TO
16344                                 ;LOCATION 4(NON-EXISTANT ADDRESS TRAP) OCCURRING. THE REGISTERS TO
16345                                 ;BE TESTED ARE:
16346                                 ;       CACHE CONTROL           17777746
16347                                 ;       MEMORY SYSTEM ERROR     17777744
16348                                 ;       HIT/MISS                17777752
16349                                 ;EACH REGISTER WILL BE ACCESSED WITH A WRITE CYCLE.
16350                                 ;
16351                                 ;BGNTST
16352                                 ;SAVE CONTENT OF LOCATION 4
16353                                 ;LET LOCATION 4 POINT TO ERROR ROUTINE
16354                                 ;INITIALIZE R TO TOP OF ADDRESS TABLE
16355                                 ;DO UNTIL FOR ALL REGISTERS
16356                                 ;.      TEST REGISTER UNDER TEST
16357                                 ;.      IF TIMEOUT DID HAPPEN
16358                                 ;.      .       ERROR
16359                                 ;.      ENDIF
16360                                 ;ENDDO
16361                                 ;RESTORE CONTENTS OF LOCATION 4
16362                                 ;EXIT   TST
16363                                 ;
16364                                 ;ADDRESS TABLE: 17777746
16365                                 ;               17777744
16366                                 ;               17777752
16367                                 ;
16368                                 ;ENDTST
16369                                 ;ERROR ROUTINE: SET TIMEOUT FLAG
16370                                 ;               RETURN
16371
16372                                 ;;*****************************************************************
      104376  000004                  TST2:   SCOPE
16373 104400  000240                          NOP
16374 104402  005737  003032                  TST     CCHPAS                  ;have done enough inclusive passes?
```

TEST - CHECK REGISTER ACCESS

```
16375 104406  001002                        BNE     99$                     ; not yet
16376
16377 104410  000240                        NOP                             ; debug aid
16378 104412  000453                        BR      TST3            ;;GO TO NEXT TEST
16379 104414  000240                99$:    NOP
16380 104416  012737  137542  000004        MOV     #TOUT,  @#4
16381 104424  013737  000004  003012        MOV     @#4,    SLOCOO          ;SAVE CONTENTS OF VECTOR 4
16382 104432  012737  104536  000004        MOV     #ERROUT,@#4             ;LET VECTOR 4 POINT TO ERROR ROUTINE
16383 104440  005001                        CLR     R1                      ;CLEAR TIMEOUT FLAG
16384 104442  005737  177746          1$:   TST     CCR                     ;READ REGISTER UNDER TEST
16385 104446  005701                        TST     R1                      ;TIMEOUT?
16386 104450  001404                        BEQ     2$
16387 104452  012737  177746  001122        MOV     #CCR,$BDADR             ;STORE LOCATION
16388 104460  104130                        ERROR   +130                    ;TIMEOUT ACCESING CCR
16389 104462  005001                2$:     CLR     R1                      ;CLEAR TIMOUT FLAG
16390 104464  005737  177744                TST     MSER                    ;READ MSER
16391 104470  005701                        TST     R1                      ;TIMEOUT ?
16392 104472  001404                        BEQ     3$
16393 104474  012737  177744  001122        MOV     #MSER,$BDADR            ;STORE LOCATION
16394 104502  104130                        ERROR   +130                    ;TIMEOUT ACCESSING MSER
16395 104504  005001                3$:     CLR     R1                      ;CLEAR TIMEOUT FLAG
16396 104506  005737  177752                TST     HITMIS                  ;ACCESS HIT/MISS
16397 104512  005701                        TST     R1                      ;TIMEOUT?
16398 104514  001404                        BEQ     4$
16399 104516  012737  177752  001122        MOV     #HITMIS,$BDADR          ;STORE LOCATION
16400 104524  104130                        ERROR   +130                    ;TIMEOUT ACCESSING HIT/MISS
16401 104526  013737  003012  000004  4$:   MOV     SLOCOO, @#4             ;RESTORE VECTOR 4
16402 104534  000402                        BR      TST3            ;;GO TO NEXT TEST
16403
16404 104536                        ERROUT:
16405 104536  005201                        INC     R1                              ;FLAG TIMEOUT
16406 104540  000002                        RTI
16407
```

TEST - CCR REGISTER BIT TEST

```
                                .SBTTL  TEST  - CCR REGISTER BIT TEST
                                ;CCR REGISTER BIT TEST - THIS TEST WILL VERIFY THAT EACH READ/WRITE BIT OF
                                ;THE CACHE CONTROL REGISTER CAN BE SET AND CLEARED INDIVIDUALLY AND THAT
                                ;BITS 15 - 11 AND BIT 8 ARE ALWAYS READ AS ZEROS.
                                ;
                                ;BGNTST
                                ;INITIALIZE GOOD DATA TO #1
                                ;CLEAR CCR
                                ;DO UNTIL ALL BITS TESTED
                                ;.      WRITE GOOD DATA TO CCR
                                ;.      READ CCR
                                ;.      IF CCR NOT EQUAL TO GOOD DATA THEN
                                ;.      .       IF CCR EQUAL TO ZERO THEN
                                ;.      .       .       IF GOOD DATA NOT EQUAL TO BIT 15-11 OR BIT 8 THEN
                                ;.      .       .       .       ERROR IN READ/WRITE BITS OF CCR
                                ;.      .       .       ENDIF
                                ;.      ENDIF
                                ;.      UPDATE TO NEXT BIT
                                ;ENDDO
                                ;ENDTST

                                ;;*****************************************************************
     104542  000004             TST3:   SCOPE
     104544  000240                     NOP
     104546  005737  003032             TST     CCHPAS                  ;have done enough inclusive passes?
     104552  001002                     BNE     99$                     ; not yet
     104554  000240                     NOP                             ; debug aid
     104556  000431                     BR      TST4            ;;GO TO NEXT TEST
     104560  000240             99$:    NOP
     104562  012701  000001             MOV     #1,     R1              ;INITIALIZE GOOD DATA TO #1
     104566  005037  177746             CLR     CCR                     ;CLEAR CCR
     104572  042737  001000  177520     BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
     104600  010137  177746     1$:     MOV     R1,     CCR             ;WRITE GOOD DATA TO CCR
     104604  023701  177746             CMP     CCR,    R1              ;IF CCR NOT EQUAL GOOD DATA
     104610  001407                     BEQ     3$                      ;THEN
     104612  005737  177746             TST     CCR                     ;IF CCR EQUAL TO ZERO
     104616  001003                     BNE     2$                      ;THEN
     104620  032701  174400             BIT     #174400,R1              ;IF GOOD DATA NE TO BIT 15-11 OR BIT 8
     104624  001001                     BNE     3$                      ;THEN
     104626  104004             2$:     ERROR   +4                      ;ERROR IN READ/WRITE BITS OF CCR
     104630  006101             3$:     ROL     R1                      ;UPDATE TO NEXT BIT
     104632  103362                     BCC     1$                      ;DO UNTIL ALL BITS TESTED
     104634  052737  001000  177520     BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
```

```
16454                                   .SBTTL  TEST  - FORCE MISS TEST
16455                                   ;FORCE MISS TEST - THIS TEST WILL VERIFY THAT ALL REFERENCES MADE
16456                                   ;WITH EITHER BIT<3> OR BIT<2> OF THE CCR SET CAUSE A CACHE MISS AND
16457                                   ;LEAVE THE CACHE ENTRY UNCHANGED. FIRST WRITE A TEST ADDRESS WITH
16458                                   ;BITS<3:2> CLEARED TO ALLOCATE CACHE AND SET A KNOWN DATA PATTERN
16459                                   ;INTO THE CACHE. THEN SET BIT<2> AND REWRITE THE SAME ADDRESS WITH
16460                                   ;NEW DATA. NEXT CLEAR BITS<3:2> AND READ THE TEST ADDRESS, THE DATA
16461                                   ;SHOULD EQUAL TO PATTERN 1. FINALLY READ THE TEST ADDRESS WITH BIT<3>
16462                                   ;SET, THE DATA SHOULD EQUAL PATTERN 2. A LAST WRITE MUST BE DONE WITH
16463                                   ;BOTH FORCE BITS CLEARED BECAUSE THE CACHE AND MAIN MEMORY HAVE
16464                                   ;DIFFERENT DATA.
16465                                   ;
16466                                   ;
16467                                   ;BGNTST
16468                                   ;WRITE TEST ADDRESS WITH PATTERN 1
16469                                   ;SET CCR BITS<3:2> = 0,1
16470                                   ;WRITE TEST ADDRESS WITH PATTERN 2
16471                                   ;CLEAR CCR BIT<3>
16472                                   ;READ TEST ADDRESS
16473                                   ;SAVE HIT/MISS REGISTER DATA
16474                                   ;COMPARE RECEIVED DATA TO PATTERN 1
16475                                   ;IF DATA NOT EQUAL THEN
16476                                   ;.      IF DATA EQUAL TO PATTERN 2 THEN
16477                                   ;.      .       IF HIT THEN
16478                                   ;.      .       .       ERROR FORCE MISS WRITES TO CACHE
16479                                   ;.      .       ELSE
16480                                   ;.      .       .       IF PARITY ERROR INDICATORS EQUAL 0 THEN
16481                                   ;.      .       .       .       ERROR FROCE MISS INVALIDATES CACHE
16482                                   ;.      .       .       ELSE
16483                                   ;.      .       .       .       IF ALL PARITY INDICATORS SET THEN
16484                                   ;.      .       .       .       .       ERROR PARITY ABORT AFTER FORCE MISS
16485                                   ;.      .       .       .       ENDIF
16486                                   ;.      .       .       .       IF TAG PARITY ERROR THEN
16487                                   ;.      .       .       .       .       ERROR  TAG PARITY ERROR AFTER FORCE MISS
16488                                   ;.      .       .       .       ELSE
16489                                   ;.      .       .       .       .       IF B0 AND  B1 ERROR THEN
16490                                   ;.      .       .       .       .       .       ERROR DATA PARITY ERROR AFTER FORCE MISS
16491                                   ;.      .       .       .       .       ENDIF
16492                                   ;.      .       .       .       .       IF B0 PARITY ERROR THEN
16493                                   ;.      .       .       .       .       .       ERROR LOW BYTE PARITY ERROR
16494                                   ;.      .       .       .       .       ENDIF
16495                                   ;.      .       .       .       .       IF B1 PARITY ERROR THEN
16496                                   ;.      .       .       .       .       .       ERROR HIGH BYTE PARITY ERROR
16497                                   ;.      .       .       .       .       ENDIF
16498                                   ;.      .       .       .       ENDIF
16499                                   ;.      .       .       ENDIF
16500                                   ;.      .       ENDIF
16501                                   ;.      ELSE
16502                                   ;.      .       ERROR IN DATA PATH
16503                                   ;.      ENDIF
16504                                   ;ENDIF
16505                                   ;SET CCR<3:2> = 1,0
16506                                   ;READ TEST ADDRESS
16507                                   ;SAVE HIT/MISS REGISTER DATA
16508                                   ;COMPARE RECEIVED DATA TO PATTERN 2
16509                                   ;IF DATA NOT EQUAL THEN
16510                                   ;.      IF RECIEVED DATA EQUAL TO PATTERN 1 THEN
```

TEST - FORCE MISS TEST

```
 16511                                  ;.       .       IF HIT THEN
 16512                                  ;.       .       .       ERROR FORCE MISS READS FROM CACHE
 16513                                  ;.       .       ELSE
 16514                                  ;.       .       .       ERROR FORCE MISS READS FROM CACHE AND MISS
 16515                                  ;.       .       ENDIF
 16516                                  ;.       ELSE
 16517                                  ;.       .       ERROR IN DATA PATH
 16518                                  ;.       ENDIF
 16519                                  ;ENDIF
 16520                                  ;CLEAR CCR<3:2>
 16521                                  ;WRITE TEST ADDRESS
 16522                                  ;ENDTST
 16523                                  ;
 16524                                  ;;*****************************************************************
      104642  000004                   TST4:   SCOPE
 16525 104644  000240                           NOP
 16526 104646  005737  003032                   TST     CCHPAS                  ;have done enough inclusive passes?
 16527 104652  001002                           BNE     99$                     ; not yet
 16528 104654  000240                           NOP                             ; debug aid
 16529 104656  000563                           BR      TST5            ;;GO TO NEXT TEST
 16530 104660  000240                   99$:    NOP
 16531 104662  013737  000114  003012           MOV     @#114,  SLOC00          ;SAVE CONTENTS OF VECTOR 114
 16532 104670  012737  105214  000114           MOV     #FMPARR,@#114           ;SETUP PARITY VECTOR TO POINT TO HANDLER
 16533 104676  005003                           CLR     R3                      ;CLEAR MSER SAVE LOCATION
 16534 104700  005037  003162                   CLR     @#TSTLOC                ;WRITE TEST LOCATION WITH PATTERN 1
 16535 104704  012737  000004  177746           MOV     #BIT02, CCR             ;SET CCR BITS<3:2> = 0,1
 16536 104712  012737  177777  003162           MOV     #177777,@#TSTLOC        ;WRITE TEST LOCATION WITH PATTERN 2
 16537 104720  012737  000200  177746           MOV     #200,   CCR             ;CLEAR CCR BIT 3 AND SET PARITY ABORTS
 16538 104726  012737  177777  001124           MOV     #177777,$GDDAT          ;SAVE DATA IN MEMORY
 16539 104734  042737  001000  177520           BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
 16540 104742  013701  003162                   MOV     @#TSTLOC,R1             ;READ TEST ADDRESS
 16541 104746  013702  177752                   MOV     @#HITMIS,R2             ;SAVE HIT/MISS DATA
 16542 104752  052737  001000  177520           BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
 16543 104760  005701                           TST     R1                      ;COMPARE RECEIVED DATA TO PATTERN 1
 16544 104762  001451                           BEQ     1$                      ;IF DATAS NOT EQUAL THEN
 16545 104764  022701  177777                   CMP     #177777,R1              ;IF DATA EQUAL TO PATTERN 2
 16546 104770  001045                           BNE     106$                    ;THEN
 16547 104772  032702  000002                   BIT     #BIT01, R2              ;IF HIT
 16548 104776  001402                           BEQ     100$                    ;THEN
 16549 105000  104005                           ERROR   +5                      ;FORCE MISS WRITES TO CACHE
 16550 105002  000441                           BR      1$                      ;ELSE
 16551 105004  032703  100340           100$:   BIT     #100340,R3              ;IF PARITY INDICATORS EQUAL 0
 16552 105010  001002                           BNE     101$                    ;THEN
 16553 105012  104006                           ERROR   +6                      ;FORCE MISS WRITE INVALIDATES CACHE
 16554 105014  000434                           BR      1$                      ;ELSE
 16555 105016  022703  100340           101$:   CMP     #100340,R3              ;IF ALL PARITY INDICATORS SET
 16556 105022  001002                           BNE     102$                    ;THEN
 16557 105024  104007                           ERROR   +7                      ;PARITY ABORT AFTER FORCE MISS
 16558 105026  000427                           BR      1$                      ;ELSE
 16559 105030  032703  000040           102$:   BIT     #BIT05, R3              ;IF TAG PARITY ERROR
 16560 105034  001402                           BEQ     103$                    ;THEN
 16561 105036  104010                           ERROR   +10                     ;TAG PARITY ERROR AFTER FORCE MISS
 16562 105040  000422                           BR      1$                      ;ELSE
 16563 105042  010304                   103$:   MOV     R3,     R4              ;COPY MSER INTO REG 4
 16564 105044  042704  177477                   BIC     #177477,R4              ;MASK FOR B0 AND B1 DATA
 16565 105050  022704  000300                   CMP     #300,   R4              ;IF B0 AND B1 DATA
 16566 105054  001002                           BNE     104$                    ;THEN
```

TEST - FORCE MISS TEST

```
16567 105056  104011                        ERROR   +11                     ;DATA PARITY ERROR AFTER FORCE MISS
16568 105060  000412                        BR      1$                      ;ELSE
16569 105062  032703  000100          104$: BIT     #100,   R3              ;IF B0 ERROR
16570 105066  001401                        BEQ     105$                    ;THEN
16571 105070  104012                        ERROR   +12                     ;LOW BYTE PARITY ERROR AFTER FORCE MISS
16572 105072  032703  000200          105$: BIT     #200,   R3              ;IF B1 ERROR
16573 105076  001403                        BEQ     1$                      ;THEN
16574 105100  104013                        ERROR   +13                     ;HIGH BYTE PARITY ERROR AFTER FORCE MISS
16575 105102  000401                        BR      1$                      ;ENDIF
16576 105104  104014                  106$: ERROR   +14                     ;ERROR DATA PATH
16577 105106  005003                  1$:   CLR     R3                      ;CLEAR MSER SAVE REGISTER
16578 105110  052737  000010  177746        BIS     #BIT03, CCR             ;SET CCR BITS <3:2> = 1,0
16579 105116  042737  001000  177520        BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
16580 105124  013701  003162                MOV     @#TSTLOC,R1             ;READ TEST LOCATION
16581 105130  013702  177752                MOV     @#HITMIS,R2             ;SAVE HIT/MISS REGISTER DATA
16582 105134  052737  001000  177520        BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
16583 105142  020127  177777                CMP     R1,     #177777         ;COMPARE RECEIVED DATA TO PATTERN 2
16584 105146  001412                        BEQ     2$                      ;IF DATAS NOT EQUAL THEN
16585 105150  005701                        TST     R1                      ;IF RECIEVED DATA EQUAL TO PATTERN 1
16586 105152  001007                        BNE     108$                    ;THEN
16587 105154  032702  000002                BIT     #BIT01, R2              ;IF HIT
16588 105160  001402                        BEQ     107$                    ;THEN
16589 105162  104015                        ERROR   +15                     ;FORCE MISS READS FROM CACHE
16590 105164  000403                        BR      2$                      ;ELSE
16591 105166  104016                  107$: ERROR   +16                     ;FORCE MISS READS FROM CACHE AND MISS
16592 105170  000401                        BR      2$                      ;ENDIF
16593 105172  104014                  108$: ERROR   +14                     ;ERROR IN DATA PATH
16594 105174  013737  003012  000114  2$:   MOV     SLOC00, @#114           ;RESTORE VECTOR 114
16595 105202  005037  177746                CLR     CCR                     ;CLEAR CCR BITS<3:2>
16596 105206  005037  003162                CLR     @#TSTLOC                ;WRITE TEST ADDRESS
16597 105212  000405                        BR      TST5            ;;GO TO NEXT TEST
16598
16599
16600 105214  011637  001122          FMPARR: MOV     (SP),$BDADR             ;SAVE ADDRESS THAT CAUSED ABORT
16601 105220  013703  177744                  MOV     MSER,   R3              ;SAVE CONTENTS OF MEMORY SYSTEM ERROR
16602 105224  000002                        RTI                             ;REGISTER AND RETURN
16603
16604
16605                                 .SBTTL  TEST  - HIT/MISS REGISTER TEST PART 1
16606                                 ;HIT/MISS REGISTER TEST PART 1 - THIS TEST WILL VERIFY THAT THE HIT/MISS
16607                                 ;REGISTER CORRECTLY LOGS HITS AND MISSES.FIRST WRITE A TEST ADDRESS WITH
16608                                 ;BITS<3:2> CLEARED TO ALLOCATE CACHE AND SET A KNOWN DATA PATTERN INTO
16609                                 ;THE CACHE. THEN SET BIT<2> AND REWRITE THE SAME ADDRESS WITH NEW DATA.
16610                                 ;NEXT CLEAR CCR BITS<3:2> AND READ THE TEST ADDRESS, THE HIT/MISS REGISTER
16611                                 ;SHOULD LOGGED A HIT. FINALLY READ THE TEST ADDRESS PLUS 20000(8) THE
16612                                 ;HIT/MISS REGISTER SHOULD HAVE LOGGED A MISS.
16613                                 ;
16614                                 ;
16615                                 ;BGNTST
16616                                 ;WRITE TEST ADDRESS WITH PATTERN 1
16617                                 ;SET CCR BITS<3:2> = 0,1
16618                                 ;WRITE TEST ADDRESS WITH PATTERN 2
16619                                 ;CLEAR CCR BIT<3>
16620                                 ;READ TEST ADDRESS
16621                                 ;IF HIT/MISS REGISTER BIT 1 NOT SET THEN
16622                                 ;.      ERROR IN RECORDING HITS IN HIT/MISS
16623                                 ;ENDIF
```

```
                                        ;READ TEST ADDRESS + 20000(8)
                                        ;IF HIT/MISS REGISTER BIT 1 SET THEN
                                        ;.      ERROR IN RECORDING HITS IN HIT/MISS
                                        ;ENDIF
                                        ;ENDTST
                                        ;
                                        ;;*****************************************************************
    105226  000004                      TST5:   SCOPE
                                        
                                                NOP
                                                TST     CCHPAS                  ;have done enough inclusive passes?
                                                BNE     99$                     ; not yet
                                                NOP                             ; debug aid
                                                BR      TST6            ;;GO TO NEXT TEST
                                        99$:    NOP
                                                MOV     @#114,  SLOC00          ;SAVE CONTENTS OF VECTOR 114
                                                MOV     #HMPARR,@#114           ;SETUP PARITY VECTOR TO POINT TO HANDLER
                                                CLR     @#TSTLOC                ;WRITE TEST LOCATION WITH PATTERN 1
                                                MOV     #BIT02, CCR             ;SET CCR BITS<3:2> = 0,1
                                                MOV     #177777,@#TSTLOC        ;WRITE TEST LOCATION WITH PATTERN 2
                                                MOV     #200,   CCR             ;CLEAR CCR BIT 3 AND SET PARITY ABORTS
                                                BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
                                                MOV     @#TSTLOC,R1             ;READ TEST ADDRESS
                                                MOV     HITMIS,RECDAT           ;STORE REGISTER
                                                BIT     #BIT02,RECDAT           ;IF HIT/MISS REGISTER BIT 1 NOT SET
                                                BNE     1$                      ;THEN
                                                ERROR   +45                     ;ERROR IN RECORDING HITS IN HIT/MISS
                                        1$:     MOV     @#TSTLOC+8192.,R1       ;READ TEST LOCATION + 20000(8)
                                                MOV     HITMIS,RECDAT           ;STORE REGISTER
                                                BIT     #BIT02,RECDAT           ;IF HIT/MISS REGISTER BIT 1 SET
                                                BEQ     2$                      ;THEN
                                                ERROR   +45                     ;ERROR IN RECORDING HITS IN HIT/MISS
                                        2$:     MOV     SLOC00, @#114           ;RESTORE VECTOR 114
                                                BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
                                                BR      TST6            ;;GO TO NEXT TEST
                                        
                                        HMPARR: MOV     MSER,   R3              ;SAVE CONTENTS OF MEMORY SYSTEM ERROR
                                                RTI                             ;REGISTER AND RETURN
```

```
16663                                   .SBTTL  TEST  - HIT/MISS REGISTER TEST
16664                                   ;HIT/MISS REGISTER TEST - THIS TEST WILL VERIFY THAT THE HIT/MISS
16665                                   ;REGISTER CORRECTLY LOGS CACHE HITS AND MISSES. IT WILL ALSO VERIFY
16666                                   ;THAT EACH BIT OF THE REGISTER IS UNIQUE. THIS WILL BE DONE BY
16667                                   ;FLOATING A ZERO THROUGH A FIELD OF ONES. THE ROTATING WILL BE DONE
16668                                   ;BY EXECUTING A TST @#HM (WHERE HM IS THE ADDRESS OF THE HIT/MISS
16669                                   ;REGISTER) AT SUCCESSIVE POSITIONS IN A SET OF EIGHT NOPS. THE READ
16670                                   ;OF THE I/O PAGE ADDRESS OF THE HIT/MISS REGISTER SHOULD CAUSE A
16671                                   ;MISS TO BE RECORDED.
16672                                   ;
16673                                   ;BGNTST
16674                                   ;INITIALIZE LOOP INDICATOR
16675                                   ;INITIALIZE EXPECTED DATA
16676                                   ;DO UNTIL LOOP INDICATOR = SEVEN
16677                                   ;.      PUT BACKGROUND DATA INTO EXECUTION BUFFER
16678                                   ;.      PUT TST INSTRUCTION INTO TEST LOCATION
16679                                   ;.      JUMP TO EXECUTE BUFFER
16680                                   ;.      IF EXPECTED DATA NE RECEIVED DATA THEN
16681                                   ;.      .       ERROR IN RECORDING HITS IN HIT/MISS
16682                                   ;.      ENDIF
16683                                   ;.      INCREMENT LOOP INDICATOR
16684                                   ;.      UPDATE EXPECTED DATA
16685                                   ;ENDDO
16686                                   ;EXIT   TST
16687                                   ;
16688                                   ;BACKGROUND DATA:NOP
16689                                   ;                NOP
16690                                   ;                NOP
16691                                   ;                NOP
16692                                   ;                NOP
16693                                   ;                NOP
16694                                   ;                NOP
16695                                   ;                NOP
16696                                   ;                MOV     @#HM,R2
16697                                   ;                RTS     PC
16698                                   ;ENDTST
16699                                   ;;*****************************************************************
    105412  000004                      TST6:   SCOPE
16700 105414  000240                              NOP
16701 105416  005737  003032                      TST     CCHPAS                  ;have done enough inclusive passes?
16702 105422  001002                              BNE     99$                     ; not yet
16703 105424  000240                              NOP                             ; debug aid
16704 105426  000511                              BR      TST7            ;;GO TO NEXT TEST
16705 105430  000240                      99$:    NOP
16706 105432  005037  003154                      CLR     LOOPIN                  ;INITIALIZE LOOP INDICATOR
16707 105436  013737  023204  001162              MOV     @#TSTLOC+20022,$TMP1    ;SAVE FOR LATER
16708 105444  012703  003164                      MOV     #TSTLOC+2,R3            ;GET ADDRESS OF EXECUTION BUFFER
16709 105450  012704  105634                      MOV     #EXPTBL,R4              ;GET ADDRESS OF EXPECTED DATA TABLE
16710 105454  012705  177752                      MOV     #HITMIS,R5              ;PUT ADDRESS OF HIT/MISS REGISTER IN GPR
16711 105460  042737  001000  177520              BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
16712 105466  023727  003154  000007  1$:       CMP     LOOPIN, #7              ;DO UNTIL LOOP INDICATOR EQUALS 7
16713 105474  001440                              BEQ     ENDHRT                  ;THEN EXIT
16714 105476  012702  000013                      MOV     #13,    R2
16715 105502  012700  105606                      MOV     #BACDAT,R0              ;PUT BACKGROUND DATA INTO
16716 105506  012701  003162                      MOV     #TSTLOC,R1              ;EXECUTION BUFFER
16717 105512  012021                      2$:     MOV     (R0)+,  (R1)+
16718 105514  077202                              SOB     R2,     2$              ;LOOP FOR TEN WORDS
```

TEST - HIT/MISS REGISTER TEST

```
16719 105516  023727  003154  000006          CMP     LOOPIN, #6              ;IS THIS LAST LOOP
16720 105524  001004                          BNE     3$                      ;IF YES THEN
16721 105526  013737  001162  023204          MOV     $TMP1,@#TSTLOC+20022    ;INVALIDATE FETCH OF "RECDAT"
16722 105534  000404                          BR      4$                      ;ELSE
16723 105536  012723  005737          3$:     MOV     #5737,  (R3)+           ;MOVE "TST" INSTRUCTION TO BUFFER
16724 105542  012713  177752                  MOV     #HITMIS,(R3)            ;MOVE HIT.MISS REG. ADD. TO BUFFER
16725 105546  004737  003162          4$:     JSR     PC,     TSTLOC          ;GO EXECUTE TEST CODE
16726 105552  021437  114150                  CMP     (R4),   RECDAT          ;IF EXPECTED DATA NOT EQUAL TO
16727 105556  001403                          BEQ     5$                      ;RECEIVED DATA THEN
16728 105560  011437  001124                  MOV     (R4),$GDDAT             ;SAVE EXPECTED PATTERN
16729 105564  104017                          ERROR   +17                     ;ERROR IN RECORDING HITS IN HIT/MISS
16730 105566  005724                  5$:     TST     (R4)+                   ;INCREMENT POINTER TO EXPECTED PATTERN
16731 105570  005237  003154                  INC     LOOPIN                  ;INCREMENT LOOP COUNTER
16732 105574  000734                          BR      1$
16733 105576                          ENDHRT:
16734 105576  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
16735 105604  000422                          BR      TST7            ;;GO TO NEXT TEST
16736
16737 105606  000240                  BACDAT: .WORD   NOP
16738 105610  000240                          .WORD   NOP
16739 105612  000240                          .WORD   NOP
16740 105614  000240                          .WORD   NOP
16741 105616  000240                          .WORD   NOP
16742 105620  000240                          .WORD   NOP
16743 105622  000240                          .WORD   NOP
16744 105624  000240                          .WORD   NOP
16745 105626  011537  114150                  MOV     (R5),RECDAT             ;SAVE CONTENTS OF HIT/MISS REGISTER
16746 105632  000207                          RTS     PC
16747
16748 105634  000077                  EXPTBL: .WORD   77
16749 105636  000037                          .WORD   37
16750 105640  000057                          .WORD   57
16751 105642  000067                          .WORD   67
16752 105644  000073                          .WORD   73
16753 105646  000075                          .WORD   75
16754 105650  000076                          .WORD   76
16755
```

```
16757                           .SBTTL  TEST  - BYTE ALLOCATION TEST
16758                           ;BYTE ALLOCATION TEST - THIS TEST WILL VERIFY THAT THE CACHE SYSTEM CORRECTLY
16759                           ;HANDLES BYTE ACCESSES. THE TEST WILL FIRST CHECK THAT A WRITE ACCESS WHICH
16760                           ;IS A MISS DOES NOT UPDATE THE CACHE. THIS WILL BE DONE BY FIRST ASSURRING
16761                           ;A MISS AT A TEST LOCATION. THEN A WRITE BYTE ACCESS WILL BE DONE. FINALLY
16762                           ;THE LOCATION WILL BE READ. THE WRITE BYTE SHOULD NOT HAVE ALLOCATED THE
16763                           ;ADDRESS. NEXT THE TEST WILL VERIFY THAT IF A WRITE BYTE ACCESS IS A HIT
16764                           ;THAT THE CACHE LOCATION IS UPDATED. THE TEST WILL FIRST WRITE A TEST LOCATION
16765                           ;WITH A PATTERN TO ALLOCATE THE ADDRESS. THEN A SECOND PATTERN WILL BE WRITTEN
16766                           ;TO THE HIGH BYTE OF THE TEST LOCATION. THE TEST LOCATION WILL THEN BE READ.
16767                           ;IF THE HIGH BYTE OF THE TEST LOCATION IS EQUAL TO THE SECOND PATTERN THEN THE
16768                           ;LOCATION WAS UPDATED. THE SECOND TEST WILL BE REPEATED TO TEST LOW BYTE
16769                           ;ACCESSES.
16770                           ;
16771                           ;BGNTST
16772                           ;SAVE VECTOR 114
16773                           ;LET VECTOR 114 POINT TO BYTE PARITY ROUTINE
16774                           ;SET PARITY ABORT BIT IN CACHE CONTROL REGISTER
16775                           ;READ TEST LOCATION + 4K
16776                           ;WRITE LOW BYTE TO TEST ADDRESS
16777                           ;READ LOW BYTE OF TEST ADDRESS
16778                           ;IF HIT/MISS REGISTER BIT 1 SET THEN
16779                           ;.      ERROR WRITE BYTE ALLOCATES THE CACHE
16780                           ;ENDIF
16781                           ;WRITE PATTERN 1 TO TEST LOCATION
16782                           ;WRITE BYTE PATTERN 2 TO TEST LOCATION+1
16783                           ;IF BIT1 NOTSETIN HIT/MISS REGISTER THEN
16784                           ;.      IF B0 PARITY ERROR THEN
16785                           ;.      .       ERROR LOW BYTE PARITY ERROR ON WRITE BYTE HIT
16786                           ;.      ELSE
16787                           ;.      .       IF B1 PARITY ERROR THEN
16788                           ;.      .       .       ERROR HIGH BYTE PARITY ERROR ON WRITE BYTE HIT
16789                           ;.      .       ELSE
16790                           ;.      .       .       WRITE BYTE HIT DOES NOT RECORD A HIT
16791                           ;.      .       ENDIF
16792                           ;.      ENDIF
16793                           ;ELSE
16794                           ;.      READ TEST LOCATION
16795                           ;.      IF RECIEVED DATA NOT EQUAL TO EXPECTED DATA THEN
16796                           ;.      .       IF BYTE DATA REVERSED THEN
16797                           ;.      .       .       ERROR BYTES REVERSED ON WRITE CYCLES
16798                           ;.      .       ELSE
16799                           ;.      .       .       IF HIGH BYTE DATA ZERO THEN
16800                           ;.      .       .       .       ERROR IN WRITING TO HIGH BYTE
16801                           ;.      .       .       ELSE
16802                           ;.      .       .       .       ERROR IN WRITING TO HIGH BYTE
16803                           ;.      .       .       ENDIF
16804                           ;.      .       ENDIF
16805                           ;.      ENDIF
16806                           ;ENDIF
16807                           ;WRITE PATTERN 1 TO TEST LOCATION
16808                           ;WRITE BYTE PATTERN 2 TO TEST LOCATION LOW BYTE
16809                           ;IF BIT1 NOTSETIN HIT/MISS REGISTER THEN
16810                           ;.      IF B0 PARITY ERROR THEN
16811                           ;.      .       ERROR LOW BYTE PARITY
16812                           ;.      ELSE
16813                           ;.      .       IF B1 PARITY ERROR THEN
```

```
16814                                   ;.      .       .       ERROR HIGH BYTE PRITY
16815                                   ;.      .       ELSE
16816                                   ;.      .       .       WRITE BYTE HIT DOES NOT RECORD A HIT
16817                                   ;.      .       ENDIF
16818                                   ;.      ENDIF
16819                                   ;ELSE
16820                                   ;.      READ TEST LOCATION
16821                                   ;.      IF RECIEVED DATA NOT EQUAL TO EXPECTED DATA THEN
16822                                   ;.      .       IF BYTE DATA REVERSED THEN
16823                                   ;.      .       .       ERROR BYTES REVERSED
16824                                   ;.      .       ELSE
16825                                   ;.      .       .       IF LOW BYTE DATA ZERO THEN
16826                                   ;.      .       .       .       ERROR IN DATA PATH
16827                                   ;.      .       .       ELSE
16828                                   ;.      .       .       .       ERROR IN DATA PATH
16829                                   ;.      .       .       ENDIF
16830                                   ;.      .       ENDIF
16831                                   ;.      ENDIF
16832                                   ;ENDIF
16833                                   ;CLEAR CACHE CONTROL REGISTER
16834                                   ;RESTORE VECTOR 114
16835                                   ;EXIT   TST
16836                                   ;
16837                                   ;BYTE PARITY ROUTINE:   SAVE MSER
16838                                   ;                       RETURN
16839                                   ;ENDTST
16840
16841                                   ;;*****************************************************************
    105652  000004                      TST7:   SCOPE
16842 105654  000240                      NOP
16843 105656  005737  003032              TST     CCHPAS                  ;have done enough inclusive passes?
16844 105662  001002                      BNE     99$                     ; not yet
16845 105664  000240                      NOP                             ; debug aid
16846 105666  000517                      BR      TST10           ;;GO TO NEXT TEST
16847 105670  000240                      99$:    NOP
16848 105672  013737  000114  003012      MOV     @#114,  SLOC00          ;SAVE VECTOR 114
16849 105700  012737  106114  000114      MOV     #BYPARR,@#114           ;LET VECTOR 114 POINT TO ABORT ROUTINE
16850 105706  005003                      CLR     R3                      ;CLEAR MSER SAVE REGISTER
16851 105710  012737  000200  177746      MOV     #BIT07, CCR             ;SET PARITY ABORT BIT IN CCR
16852 105716  005737  023162              TST     TSTLOC+8192.            ;READ TEST LOCATION + 4K TO CAUSE MISS
16853 105722  042737  001000  177520      BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
16854 105730  105037  003162              CLRB    TSTLOC                  ;WRITE LOW BYTE OF TEST LOCATION
16855 105734  105737  003162              TSTB    TSTLOC                  ;READ LOW BYTE OF TEST LOCATION
16856 105740  013737  177752  114150      MOV     HITMIS,RECDAT           ;STORE REGISTER
16857 105746  032737  000004  114150      BIT     #BIT02,RECDAT           ;IF BIT1 OF HIT/MISS REGISTER SET
16858 105754  001401                      BEQ     1$                      ;THEN
16859 105756  104020                      ERROR   +20                     ;WRITE BYTE ALLOCATES CACHE
16860 105760  005037  003162              1$:     CLR     TSTLOC          ;WRITE PATTERN 1 TO TEST LOCATION
16861 105764  152737  000377  003163      BISB    #377,   @#TSTLOC+1      ;WRITE PATTERN 2 TO HIGH BYTE
16862 105772  013737  177752  114150      MOV     HITMIS,RECDAT           ;STORE REGISTER
16863 106000  032737  000004  114150      BIT     #BIT02,RECDAT           ;IF BIT1 NOTSETIN HIT/MISS REGISTER
16864 106006  001002                      BNE     2$                      ;THEN
16865 106010  104021                      101$:   ERROR   +21             ;WRITE BYTE HIT DOES NOT RECORD HIT
16866 106012  000405                      BR      3$                      ;ELSE
16867 106014  022737  177400  003162      2$:     CMP     #177400,@#TSTLOC        ;IF TEST LOCATION NOT EQUAL TO
16868 106022  001401                      BEQ     3$                      ;PATTERN 2 THEN
16869 106024  104022                      ERROR   +22                     ;BYTES REVERSED ON WRITE CYCLES
```

TEST - BYTE ALLOCATION TEST

```
16870 106026  005037  003162          3$:     CLR     TSTLOC                  ;WRITE PATTERN 1 TO TEST LOCATION
16871 106032  152737  000377  003162          BISB    #377,   @#TSTLOC        ;WRITE PATTERN 2 TO LOW BYTE
16872 106040  013737  177752  114150          MOV     HITMIS,RECDAT           ;STORE REGISTER
16873 106046  032737  000004  114150          BIT     #BIT02,RECDAT           ;IF BIT1 NOTSETIN HIT/MISS REGISTER
16874 106054  001001                          BNE     4$                      ;THEN
16875 106056  104012                          ERROR   ^12                     ;LOW BYTE PARITY ERROR ON WRITE BYTE HIT
16876 106060  022737  000377  003162  4$:     CMP     #377,   @#TSTLOC        ;IF TEST LOCATION NOT EQUAL TO
16877 106066  001401                          BEQ     5$                      ;PATTERN 2 THEN
16878 106070  104022                          ERROR   ^22                     ;BYTES REVERSED ON WRITE CYCLES
16879 106072  005037  177746          5$:     CLR     CCR                     ;CLEAR CCR BEFORE EXIT
16880 106076  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
16881 106104  013737  003012  000114          MOV     SLOC00, @#114           ;RESTORE VECTOR 114
16882 106112  000405                          BR      TST10           ;;GO TO NEXT TEST
16883
16884
16885 106114  011637  001122          BYPARR: MOV     (SP),   $BDADR          ;SAVE ADDRESS THAT CAUSE ABORT
16886 106120  013703  177744                  MOV     MSER,   R3              ;SAVE CONTENTS OF MSER
16887 106124  000002                          RTI                             ;RETURN
16888
```

TEST - PDR BIT15 (BYPASS) TEST

```
16890                                   .SBTTL  TEST  - PDR BIT15 (BYPASS) TEST
16891                                   ;PDR BIT15 (BYPASS) TEST - THIS TEST WILL VERIFY THAT WHEN BIT<15> IS SET
16892                                   ;IN A PDR AND AN ACCESS IS MADE TO A LOCATION MAPPED BY THE SELECTED PDR
16893                                   ;THAT WOULD NORMALLY CAUSE A CACHE ACCESS, THE CACHE IS BYPASSED (I.E. THE
16894                                   ;CACHE LOCATION IS INVALIDATED AND A MAIN MEMORY IS READ. THIS WILL BE DONE
16895                                   ;BY FIRST WRITING A TEST LOCATION WITH A KNOWN PATTERN TO ALLOCATE THE ADDRESS.
16896                                   ;THEN THE LOCATION WILL BE WRITTEN AGAIN WITH THE MMU ENABLED AND BIT <15> SET
16897                                   ;IN THE CONTROLING PDR WITH A NEW PATTERN. THE LOCATION WILL THEN BE READ
16898                                   ;AND A MISS SHOULD BE LOGGED IN THE HIT/MISS REGISTER. THIS READ WILL ALSO
16899                                   ;BRING VALID DATA INTO CACHE FROM MEMORY. THE MMU WILL AGAIN BE ENABLED
16900                                   ;WITH BIT <15> SET AND A READ CYCLE DONE. THIS SHOULD INVALIDATE THE CACHE.
16901                                   ;NEXT THE MMU WILL BE DISABLED AND THE LOCATION READ FOR A THIRD TIME. THIS
16902                                   ;WILL BE DONE WITH BIT<15> STILL SET. A MISS SHOULD ALSO BE LOGGED. LASTLY
16903                                   ;BIT<15> WILL BE CLEARED AND THE MMU ENABLED AND THE LOCATION AGAIN READ.
16904                                   ;THIS TIME A CACHE HIT SHOULD BE LOGGED.
16905                                   ;
16906                                   ;
16907                                   ;BGNTST
16908                                   ;SETUP MMU REGISTERS
16909                                   ;WRITE TEST LOCATION WITH PATTERN 1
16910                                   ;SET BIT<15> IN PDR FOR TEST LOCATION
16911                                   ;ENABLE MMU
16912                                   ;WRITE TEST LOCATION WITH PATTERN 2
16913                                   ;DISABLE MMU AND CLEAR BIT<15> IN PDR
16914                                   ;READ TEST LOCATION
16915                                   ;IF HIT/MISS REGISTER BIT<1> SET THEN
16916                                   ;.      ERROR CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE
16917                                   ;ENDIF
16918                                   ;SET BIT<15> IN PDR FOR TEST LOCATION
16919                                   ;ENABLE MMU
16920                                   ;READ TEST LOCATION (SHOULD INVALIDATE CACHE)
16921                                   ;DISABLE MMU
16922                                   ;READ TEST LOCATION
16923                                   ;IF HIT/MISS REGISTER BIT<1> SET THEN
16924                                   ;.      ERROR CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE
16925                                   ;ENDIF
16926                                   ;CLEAR BIT<15> IN PDR
16927                                   ;TURN ON MMU
16928                                   ;READ TEST LOCATION
16929                                   ;IF HIT/MISS REGISTER BIT<1> NOT SET THEN
16930                                   ;.      ERROR IN RECORDING HITS IN HIT/MISS
16931                                   ;ENDIF
16932                                   ;TURN OFF MMU
16933                                   ;ENDTST
16934                                   ;
16935                                   ;;*****************************************************************
     106126  000004                     TST10:  SCOPE
16936 106130  000240                             NOP
16937 106132  005737  003032                     TST     CCHPAS                  ;have done enough inclusive passes?
16938 106136  001002                             BNE     99$                     ; not yet
16939 106140  000240                             NOP                             ; debug aid
16940 106142  000512                             BR      TST11           ;;GO TO NEXT TEST
16941 106144  000240                     99$:    NOP
16942 106146  004737  136574                     JSR     PC,     INITMM          ;SETUP MEMORY MANAGEMEMT REGISTERS
16943 106152  005037  003162                     CLR     @#TSTLOC                ;WRITE TEST LOCATION WITH PATTERN 1
16944 106156  012737  177777  001124             MOV     #177777,$GDDAT          ;SAVE DATA IN MEMORY
16945 106164  052737  100000  172300             BIS     #BIT15, KIPDR0          ;SET BIT15 IN PDR FOR TEST LOCATION
```

```
16946 106172  005237  177572                    INC     SR0                     ;TURN ON MEMORY MANAGEMENT
16947 106176  012737  177777  003162            MOV     #177777,@#TSTLOC        ;WRITE TEST LOCATION WITH PATTERN 2
16948 106204  005037  177572                    CLR     SR0                     ;TURN OFF MEMORY MANAGEMENT
16949 106210  042737  100000  172300            BIC     #BIT15, KIPDR0          ;CLEAR BIT15 IN PDR FOR TEST LOCATION
16950 106216  042737  001000  177520            BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
16951 106224  013701  003162                    MOV     @#TSTLOC,R1             ;READ TEST LOCATION
16952 106230  013737  177752  114150            MOV     HITMIS,RECDAT           ;SAVE HIT/MISS REGISTER
16953 106236  032737  000004  114150            BIT     #BIT02,RECDAT           ;IF HIT/MISS REGISTER BIT 1 SET
16954 106244  001401                            BEQ     2$                      ;THEN
16955 106246  104023                            ERROR   +23                     ;CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE
16956 106250  052737  100000  172300  2$:       BIS     #BIT15, KIPDR0          ;SET BIT15 IN PDR FOR TEST LOCATION
16957 106256  005237  177572                    INC     SR0                     ;TURN ON MEMORY MANAGEMENT
16958 106262  005737  003162                    TST     TSTLOC                  ;READ TEST LOCATION
16959 106266  042737  100000  172300            BIC     #BIT15, KIPDR0          ;CLEAR BIT 15 IN PDR
16960 106274  005037  177572                    CLR     SR0                     ;TURN OFF MMU
16961 106300  013701  003162                    MOV     @#TSTLOC,R1             ;READ TEST LOCATION
16962 106304  013737  177752  114150            MOV     HITMIS,RECDAT           ;STORE HIT/MISS TO REGISTER 2
16963 106312  032737  000004  114150            BIT     #BIT02,RECDAT           ;IF HIT/MISS REGISTER BIT 1 SET
16964 106320  001401                            BEQ     3$                      ;THEN
16965 106322  104023                            ERROR   +23                     ;CONDITIONAL BYPASS DOESN'T INVALIDATE 0 CACHE
16966 106324  005237  177572          3$:       INC     SR0                     ;TURN ON MEMORY MANAGEMENT
16967 106330  005737  003162                    TST     @#TSTLOC                ;READ TEST LOCATION ONE MORE TIME
16968 106334  013737  177752  114150            MOV     HITMIS,RECDAT           ;STORE HIT/MISS TO REGISTER 2
16969 106342  032737  000004  114150            BIT     #BIT02,RECDAT           ;IF HIT/MISS REGISTER BIT 1 NOT SET
16970 106350  001001                            BNE     4$                      ;THEN
16971 106352  104045                            ERROR   +45                     ;ERROR IN RECORDING HITS IN HIT/MISS
16972 106354  042737  000001  177572  4$:       BIC     #BIT00, SR0             ;TURN OFF MMU
16973 106362  052737  001000  177520            BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
16974
```

```
16976                                   .SBTTL  TEST  - FLUSH CACHE TEST
16977                                   ;FLUSH CACHE TEST - THIS TEST WILL VERIFY THAT WHEN CCR BIT<8> IS
16978                                   ;SET, THE ENTIRE CACHE IS INVALIDATED. FIRST 8K BYTES OF ADDRESSES
16979                                   ;WILL BE READ TO ALLOCATE CACHE. THEN THE CACHE WILL BE FLUSHED.
16980                                   ;THE SAME SET OF ADDRESS WILL THEN BE READ AND THE HIT/MISS REGISTER
16981                                   ;CHECKED AFTER EACH READ FOR A MISS TO BE LOGGED.
16982                                   ;
16983                                   ;BGNTST
16984                                   ;GET FIRST ADDRESS OF 8KB BUFFER
16985                                   ;INITIALIZE LOOP COUNTER TO 4KWORD COUNT
16986                                   ;DO UNTIL LOOP COUNTER = 0
16987                                   ;.      READ @ADDRESS+
16988                                   ;ENDDO
16989                                   ;GET FIRST ADDRESS OF 8KB BUFFER
16990                                   ;INITIALIZE LOOP COUNTER TO 2KWORD COUNT
16991                                   ;INITIALIZE MISS COUNT TO 4KWORD COUNT
16992                                   ;FLUSH CACHE
16993                                   ;DO UNTIL LOOP COUNTER = 0
16994                                   ;.      READ @ADDRESS+
16995                                   ;.      ICREMENT ADDRESS TO READ EVERY 2WORDS
16996                                   ;.      IF HIT/MISS REGISTER BIT<1> SET THEN
16997                                   ;.      .       DECREMENT MISS COUNT
16998                                   ;.      ENDIF
16999                                   ;ENDDO
17000                                   ;GET FIRST ADDRESS OF 8KB BUFFER
17001                                   ;INITIALIZE LOOP COUNTER TO 4KWORD COUNT
17002                                   ;DO UNTIL LOOP COUNTER = 0
17003                                   ;.      READ @ADDRESS+
17004                                   ;ENDDO
17005                                   ;GET LAST ADDRESS OF 8KB BUFFER
17006                                   ;INITIALIZE LOOP COUNTER TO 2KWORD COUNT
17007                                   ;FLUSH CACHE
17008                                   ;DO UNTIL LOOP COUNTER = 0
17009                                   ;.      READ @ADDRESS-
17010                                   ;.      DECREMENT ADDRESS TO READ EVERY 2WORDS
17011                                   ;.      IF HIT/MISS REGISTER BIT<1> SET THEN
17012                                   ;.      .       DECREMENT MISS COUNT
17013                                   ;.      ENDIF
17014                                   ;ENDDO
17015                                   ;IF MISS COUNT NOT EQUAL TO 4096. THEN
17016                                   ;.      ERROR HITS RECORDED AFTER FLUSHING CACHE
17017                                   ;ENDIF
17018                                   ;ENDTST
17019                                   ;
17020                                   ;;*****************************************************************
    106370  000004                      TST11:  SCOPE
17021 106372  000240                          NOP
17022 106374  005737  003032                   TST     CCHPAS                  ;have done enough inclusive passes?
17023 106400  001002                          BNE     99$                     ; not yet
17024 106402  000240                          NOP                             ; debug aid
17025 106404  000517                          BR      TST12           ;;GO TO NEXT TEST
17026 106406  000240                  99$:    NOP
17027 106410  004737  136574                   JSR     PC,     INITMM          ;INITIALIZE MMU
17028 106414  012737  001600  172354           MOV     #1600,KIPAR6            ;LET PAR6 POINT TO LOW 28K
17029 106422  012701  140000                   MOV     #140000,R1              ;GET ADDRESS OF 8K BYTE BUFFER
17030 106426  005237  177572                   INC     SR0                     ;ENABLE MMU
17031 106432  012702  010000                   MOV     #4096., R2              ;INIT LOOP COUNTER TO 4K WORD COUNT
```

TEST - FLUSH CACHE TEST

```
17032 106436  005721                1$:   TST   (R1)+              ;READ ADDRESS TO ALLOCATE CACHE
17033 106440  077202                      SOB   R2,     1$          ;DO UNTIL ALL LOCATIONS ALLOCATED
17034 106442  012701  140000              MOV   #140000,R1          ;GET ADDRESS OF 8K BYTE BUFFER
17035 106446  012702  004000              MOV   #2048., R2          ;INIT LOOP COUNTER TO 2K WORD COUNT
17036 106452  012703  010000              MOV   #4096., R3          ;INITIALIZE MISS COUNT TO 4K
17037 106456  012737  000400  177746      MOV   #400,   CCR         ;FLUSH CACHE
17038 106464  042737  001000  177520      BIC   #1000,BCSR          ;DISABLE HALT ON BREAK
17039 106472  005721                2$:   TST   (R1)+               ;READ ADDRESS
17040 106474  013737  177752  114150      MOV   @#HITMIS,RECDAT     ;STORE REGISTER
17041 106502  032737  000004  114150      BIT   #BIT02,RECDAT       ;IF HIT/MISS REGISTER NOT EQUAL
17042 106510  001401                      BEQ   3$                  ;TO ZERO THEN
17043 106512  005303                      DEC   R3                  ;DECREMENT MISS COUNT
17044 106514  062701  000002          3$:   ADD   #2,R1               ;DO IN 2 WORD INCREMENTS (PMI MEMORY)
17045 106520  077214                      SOB   R2,     2$          ;LOOP UNTIL ENTIRE CACHE CHECKED
17046 106522  052737  001000  177520      BIS   #1000,BCSR          ;ENABLE HALT ON BREAK
17047 106530  012702  010000              MOV   #4096.,R2           ;DO FOR 4K
17048 106534  012701  140000              MOV   #140000,R1          ;STARTING WITH 0
17049 106540  005721                4$:   TST   (R1)+               ;ALLOCATE IN CACHE
17050 106542  077202                      SOB   R2,4$               ;DO FOR 4K
17051 106544  012702  004000              MOV   #2048.,R2           ;DO FOR THE 2ND HALF
17052 106550  012701  160000              MOV   #160000,R1          ;START WITH LAST LOCATION
17053 106554  012737  000400  177746      MOV   #400,CCR            ;FLUSH CACHE
17054 106562  042737  001000  177520      BIC   #1000,BCSR          ;DISABLE HALT ON BREAK
17055 106570  005741                5$:   TST   -(R1)               ;READ ADDRESS
17056 106572  013737  177752  114150      MOV   @#HITMIS,RECDAT     ;STORE REGISTER
17057 106600  032737  000004  114150      BIT   #BIT02,RECDAT       ;IF HIT/MISS REGISTER NOT EQUAL
17058 106606  001401                      BEQ   6$                  ;TO ZERO THEN
17059 106610  005303                      DEC   R3                  ;DECREMENT MISS COUNT
17060 106612  162701  000002          6$:   SUB   #2,R1               ;DO IN 2 WORD DECREMENTS
17061 106616  077214                      SOB   R2,5$               ;DO FOR 2K WORDS
17062 106620  052737  001000  177520      BIS   #1000,BCSR          ;ENABLE HALT ON BREAK
17063 106626  005037  177572              CLR   SR0                 ;TURN OFF MMU
17064 106632  022703  010000              CMP   #4096., R3          ;IF MISS COUNT NOT EQUAL TO 4K WORDS
17065 106636  001401                      BEQ   8$                  ;THEN
17066 106640  104024                      ERROR +24                 ;HITS RECORDED AFTER FLUSHING CACHE
17067 106642                        8$:
      106642  000400                      BR    TST12               ;;GO TO NEXT TEST
17068
```

```
                                .SBTTL  TEST  - UNCONDITIONAL BYPASS TEST
                                ;UNCONDITIONAL BYPASS TEST - THIS TEST WILL VERIFY THAT WHEN CCR
                                ;BIT<9> IS SET, A MEMORY REFERENCE IS FORCED TO MAIN MEMORY. THIS
                                ;WILL ALSO VERIFY THAT READ AND WRITE HIT INVALIDATE THE CACHE
                                ;LOCATIONS WHEN BYPASS IS SET.
                                ;
                                ;
                                ;BGNTST
                                ;
                                ;READ TEST LOCATIONS TO SETUP POSSIBILITY OF HIT
                                ;SET CCR BIT<9>
                                ;
                                ;READ FIRST TEST LOCATION
                                ;IF HIT/MISS REGISTER BIT<1> NOT SET THEN
                                ;.      ERROR IN RECORDING HITS IN HIT/MISS
                                ;ENDIF
                                ;
                                ;WRITE SECOND TEST LOCATION
                                ;IF HIT/MISS REGISTER BIT<1> NOT SET THEN
                                ;.      ERROR IN RECORDING HITS IN HIT/MISS
                                ;ENDIF
                                ;CLEAR CCR BIT<9>
                                ;
                                ;READ FIRST LOCATION
                                ;IF HIT/MISS REGISTER BIT<1> SET THEN
                                ;.      ERROR BYPASS DOESN'T INVALIDATE CACHE
                                ;ENDIF
                                ;
                                ;READ SECOND LOCATION
                                ;IF HIT/MISS REGISTER BIT<1> SET THEN
                                ;.      ERROR BYPASS DOESN'T INVALIDATE CACHE
                                ;ENDIF
                                ;
                                ;ENDTST
                                ;
                                ;
                                ;;*****************************************************************
    106644  000004              TST12:  SCOPE
17108 106646  000240                    NOP
17109 106650  005737  003032            TST     CCHPAS                  ;have done enough inclusive passes?
17110 106654  001002                    BNE     99$                     ; not yet
17111 106656  000240                    NOP                             ; debug aid
17112 106660  000471                    BR      TST13           ;;GO TO NEXT TEST
17113 106662  000240              99$:    NOP
17114
17115 106664  005737  003162            TST     @#TSTLOC                ;ALLOCATE FIRST TEST LOCATION
17116 106670  005737  003164            TST     @#TSTLOC+2              ;ALLOCATE SECOND TEST LOCATION
17117 106674  052737  001000  177746    BIS     #BIT09, CCR             ;SET CCR BIT 9 (CACHE BYPASS)
17118 106702  042737  001000  177520  1$:     BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
17119 106710  005737  003162            TST     @#TSTLOC                ;READ FIRST TEST LOCATION
17120 106714  013737  177752  114150    MOV     HITMIS,RECDAT           ;STORE HIT/MISS TO REGISTER 2
17121 106722  032737  000004  114150    BIT     #BIT02,RECDAT           ;IF HIT/MISS REG. BIT 1 NOT SET
17122 106730  001001                    BNE     2$                      ;THEN
17123 106732  104045                    ERROR   +45                     ;ERROR IN RECORDING HITS IN HIT/MISS WITH CCR<9>=1
17124 106734  005037  003164    2$:     CLR     @#TSTLOC+2              ;WRITE SECOND LOCATION
17125 106740  013737  177752  114150    MOV     HITMIS,RECDAT           ;STORE HIT/MISS TO REGISTER 2
```

TEST - UNCONDITIONAL BYPASS TEST

```
 17126 106746  032737  000004  114150          BIT     #BIT02,RECDAT           ;IF HIT/MISS REG. BIT 1 NOT SET
 17127 106754  001001                          BNE     3$                      ;THEN
 17128 106756  104045                          ERROR   +45                     ;ERROR IN RECORDING HITS IN HIT/MISS WITH CCCR<9>=1
 17129 106760  005737  003162          3$:     TST     @#TSTLOC                ;READ FIRST TEST LOCATION
 17130 106764  013737  177752  114150          MOV     HITMIS,RECDAT           ;STORE HIT/MISS TO REGISTER 2
 17131 106772  032737  000004  114150          BIT     #BIT02,RECDAT           ;IF HIT/MISS REG. BIT 1 SET
 17132 107000  001401                          BEQ     4$                      ;THEN
 17133 107002  104025                          ERROR   +25                     ;BYPASS DOESN'T INVALIDATE CACHE
 17134 107004  005037  003164          4$:     CLR     @#TSTLOC+2              ;WRITE SECOND LOCATION
 17135 107010  013737  177752  114150          MOV     HITMIS,RECDAT           ;STORE HIT/MISS TO REGISTER 2
 17136 107016  032737  000004  114150          BIT     #BIT02,RECDAT           ;IF HIT/MISS REG. BIT 1 SET
 17137 107024  001401                          BEQ     5$                      ;THEN
 17138 107026  104025                          ERROR   +25                     ;BYPASS DOESN'T INVALIDATE CACHE
 17139 107030  042737  001000  177746  5$:     BIC     #BIT09,CCR              ;RESTORE CCR
 17140 107036  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
 17141
```

TEST - WRITE WRONG DATA PARITY TEST

```
17143                                   .SBTTL  TEST  - WRITE WRONG DATA PARITY TEST
17144                                   ;WRITE WRONG DATA PARITY TEST - THIS TEST WILL VERIFY THAT WHEN CCR
17145                                   ;BIT<6> = 1 AND CCR BITS<7,0> = 0,1, A READ MISS OCCURS AFTER A
17146                                   ;WRITE. THE WRITE WITH CCR BIT<6> = 1 TO A LOCATION WILL CAUSE A
17147                                   ;CACHE UPDATE AND WRONG PARITY TO BE WRITTEN SO WHEN THE LOCATION
17148                                   ;IS READ INSTEAD OF A HIT BEING RECORDED THE CACHE PARITY ERROR
17149                                   ;WILL CAUSE A CACHE MISS. THIS TEST WILL BE DONE A BYTE AT A TIME.
17150                                   ;
17151                                   ;BGNTST
17152                                   ;SAVE CONTENTS OF VECTOR 114
17153                                   ;LET VECTOR 114 POINT TO ABORT ROUTINE
17154                                   ;CLEAR MSER
17155                                   ;IF MSER NOT CLEAR THEN
17156                                   ;.      ERROR MSER DOESN'T CLEAR ON WRITE REFERENCE
17157                                   ;ENDIF
17158                                   ;WRITE TEST LOCATION
17159                                   ;SET BITS<6,0> IN CCR
17160                                   ;WRITE TEST LOCATION LOW BYTE
17161                                   ;CLEAR CCR BIT<6> (WRITE WRONG DATA PARITY)
17162                                   ;INITIALIZE ERROR INDICATORS
17163                                   ;READ TEST LOCATION LOW BYTE
17164                                   ;IF BIT<1> SET IN HIT/MISS REGISTER THEN
17165                                   ;.      PARITY ERROR DOESN'T CAUSE MISS
17166                                   ;ELSE
17167                                   ;.      IF MSER BITS <7:5> ZERO THEN
17168                                   ;.      .       PARITY ERROR DOESN'T SET MSER PROPERLY
17169                                   ;.      ELSE
17170                                   ;.      .       SET ERROR IN LOW BYTE INDICATOR
17171                                   ;.      ENDIF
17172                                   ;.      ENDIF
17173                                   ;ENDIF
17174                                   ;CLEAR MSER
17175                                   ;IF MSER NOT CLEAR THEN
17176                                   ;.      .       ERROR MSER DOESN'T CLEAR ON WRITE REFERENCE
17177                                   ;ENDIF
17178                                   ;SET CCR BIT<6>
17179                                   ;WRITE TEST LOCATION HIGH BYTE
17180                                   ;CLEAR CCR BIT<6>
17181                                   ;READ TEST LOCATION HIGH BYTE
17182                                   ;IF BIT<1> SET IN HIT/MISS REGISTER THEN
17183                                   ;.      IF MSER BITS<7:5> NOT SET THEN
17184                                   ;.      .       PARITY ERROR DOESN'T CAUSE A MISS
17185                                   ;.      ELSE
17186                                   ;.      .       SET ERROR IN HIGH BYTE INDICATOR
17187                                   ;.      ENDIF
17188                                   ;ELSE
17189                                   ;.      IF MSER BITS <7:5> ZERO THEN
17190                                   ;.      .       PARITY ERROR DOESN'T SET MSER
17191                                   ;.      ENDIF
17192                                   ;ENDIF
17193                                   ;RESTORE CONTENTS OF VECTOR 114
17194                                   ;IF ERROR INDICATORS SET THEN
17195                                   ;.      IF ERROR IN BOTH BYTES THEN
17196                                   ;.      .       PARITY ERROR IGNORED
17197                                   ;.      ELSE
17198                                   ;.      .       IF ERROR IN LOW BYTE THEN
17199                                   ;.      .       .       LOW BYTE PARITY ERROR IGNORED
```

```
                                        ;.      .       ELSE
                                        ;.      .               HIGH BYTE PARITY ERROR IGNORED
                                        ;.      .       ENDIF
                                        ;.      ENDIF
                                        ;ENDIF
                                        ;EXIT   TST
                                        ;
                                        ;DATA PARITY ABORT ROUTINE:
                                        ;                               ILLEGAL PARITY INTERRUPT
                                        ;                               RETURN
                                        ;
                                        ;ENDTST
                                        ;;*****************************************************************
 107044  000004                         TST13:  SCOPE
17213 107046  000240                            NOP
17214 107050  005737  003032                      TST     CCHPAS                  ;have done enough inclusive passes?
17215 107054  001002                            BNE     99$                     ; not yet
17216 107056  000240                            NOP                             ; debug aid
17217 107060  000537                            BR      TST14           ;;GO TO NEXT TEST
17218 107062  000240                    99$:    NOP
17219 107064  013737  000114  003012            MOV     @#114,  SLOC00          ;SAVE CONTENTS OF VECTOR 114
17220 107072  012737  107350  000114            MOV     #DAPABO,@#114           ;LET VECTOR POINT TO ABORT ROUTINE
17221 107100  005037  177744                    CLR     MSER                    ;CLEAR MSER
17222 107104  005737  177744                    TST     MSER                    ;IF MSER NOT CLEAR
17223 107110  001401                            BEQ     1$                      ;THEN
17224 107112  104026                            ERROR   .+26                    ;MSER DOES NOT CLEAR ON WRITE REFERENCE
17225 107114  005037  003162            1$:     CLR     @#TSTLOC                ;WRITE TEST LOCATION TO ALLOCATE CACHE
17226 107120  042737  001000  177520            BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
17227 107126  012737  000101  177746            MOV     #101,   CCR             ;SET BITS<6,0> IN CCR
17228 107134  112737  000377  003162            MOVB    #377,   TSTLOC          ;WRITE LOW BYTE WITH BAD PARITY
17229 107142  042737  000100  177746            BIC     #BIT06, CCR             ;CLEAR WRITE WRONG DATA PARITY BIT
17230 107150  005002                            CLR     R2                      ;CLEAR ERROR INDICATORS
17231 107152  105737  003162                    TSTB    @#TSTLOC                ;READ LOW BYTE OF TEST LOCATION
17232 107156  013703  177752                    MOV     HITMIS, R3              ;SAVE HIT/MISS
17233 107162  032703  000004                    BIT     #BIT02, R3              ;IF BIT 1 SET IN HIT/MISS REGISTER
17234 107166  001402                            BEQ     2$                      ;THEN
17235 107170  104027                            ERROR   .+27                    ;PARITY ERROR DON'T CAUSE A MISS
17236 107172  000405                            BR      3$                      ;ELSE
17237 107174  032737  000340  177744    2$:     BIT     #340,   MSER            ;IF MSER BIT<7:5> NOT ZERO
17238 107202  001001                            BNE     3$                      ;THEN
17239 107204  104030                            ERROR   .+30                    ;PARITY ERROR DON'T SET MSER WITH CCR<7>=0
17240 107206  005037  177744            3$:     CLR     MSER                    ;CLEAR MSER
17241 107212  005737  177744                    TST     MSER                    ;IF MSER NOT CLEAR
17242 107216  001401                            BEQ     4$                      ;THEN
17243 107220  104026                            ERROR   .+26                    ;MSER DOES NOT CLEAR ON WRITE REFERENCE
17244 107222  052737  000100  177746    4$:     BIS     #BIT06, CCR             ;SET CCR BIT 6 (WRITE WRONG PARITY)
17245 107230  112737  000377  003163            MOVB    #377,   @#TSTLOC+1      ;WRITE HIGH BYTE OF TEST LOCATION
17246 107236  042737  000100  177746            BIC     #BIT06, CCR             ;CLEAR WRITE WRONG PARITY BIT
17247 107244  105737  003163                    TSTB    @#TSTLOC+1              ;READ HIGH BYTE OF TEST LOCATION
17248 107250  013737  177752  114150            MOV     HITMIS,RECDAT           ;SAVE HIT/MISS
17249 107256  032737  000004  114150            BIT     #BIT02,RECDAT           ;IF BIT 1 SET IN HIT/MISS REGISTER
17250 107264  001402                            BEQ     5$                      ;THEN
17251 107266  104027                            ERROR   .+27                    ;PARITY ERROR DON'T CAUSE A MISS
17252 107270  000405                            BR      6$                      ;ELSE
17253 107272  032737  000340  177744    5$:     BIT     #340,   MSER            ;IF BITS <7:5> ZERO
17254 107300  001001                            BNE     6$                      ;THEN
17255 107302  104030                            ERROR   .+30                    ;PARITY ERROR DON'T SET MSER WITH CCR<7>=0
```

TEST - WRITE WRONG DATA PARITY TEST

```
 17256 107304  005037  177746          6$:     CLR     CCR                     ;CLEAR CCR BEFORE EXIT
 17257 107310  013737  003012  000114          MOV     SLOC00, @#114           ;RESTORE CONTENTS OF VECTOR 114
 17258 107316  032702  000003                  BIT     #3,     R2              ;IF ERROR INDICATORS SET
 17259 107322  001401                          BEQ     7$                      ;THEN
 17260 107324  104031                          ERROR   +31                     ;PARITY ERROR IGNORED
 17261 107326  005037  177746          7$:     CLR     CCR                     ;CLEAR CCR
 17262 107332  013737  003012  000114          MOV     SLOC00,@#114            ;RESTORE PARITY TRAP
 17263 107340  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
 17264 107346  000404                          BR      TST14           ;;GO TO NEXT TEST
 17265
 17266
 17267 107350  011637  001122          DAPABO: MOV     (SP),   $BDADR          ;SAVE ADDRESS THAT CAUSED ABORT
 17268 107354  104007                          ERROR   +7                      ;ILLEGAL PARITY INTERRUPT
 17269 107356  000002                          RTI
 17270
```

```
                                        .SBTTL  TEST  - WRITE WRONG TAG PARITY
                                        ;WRITE WRONG TAG PARITY - THIS TEST WILL VERIFY THAT A READ MISS
                                        ;OCCURS AFTER A WRITE WILL CCR<10> = 1 AND CCR<7,0> = 0,1. THE WRITE
                                        ;TO THE LOCATION WILL CAUSE A CACHE UPDATE BUT THE TAG WILL BE
                                        ;WRITTEN WITH THE WRONG PARITY. WHEN THE LOCATION IS READ INSTEAD
                                        ;OF A CACHE HIT OCCURRING THE PARITY ERROR SHOULD CAUSE A CACHE
                                        ;MISS.
                                        ;
                                        ;BGNTST
                                        ;SAVE CONTENTS OF VECTOR 114
                                        ;LET VECTOR 114 POINT TO ABORT ROUTINE
                                        ;CLEAR MSER
                                        ;SET WRITE WRONG TAG PARITY BIT<10>
                                        ;WRITE TEST LOCATION
                                        ;CLEAR CCR BIT<10> AND SET ABORT DISABLE BIT<0>
                                        ;READ TEST LOCATION
                                        ;IF BIT<1> SET IN HIT/MISS REGISTER THEN
                                        ;.      IF MSER BITS <7:5> SET THEN
                                        ;.              PARITY ERROR DOESN'T CAUSE MISS
                                        ;.      ELSE
                                        ;.              PARITY ERROR DOESN'T SET MSER WITH CCR<7>=0
                                        ;.      ENDIF
                                        ;ENDIF
                                        ;SET WRITE WRONG TAG PARITY BIT<10>
                                        ;WRITE TO A BYTE OF A TEST LOCATION
                                        ;SAVE HIT/MISS REGISTER
                                        ;CLEAR WRITE WRONG TAG PARITY BIT
                                        ;IF BIT<1> NOT SET IN HIT/MISS REGISTER THEN
                                        ;.      ERROR
                                        ;ENDIF
                                        ;RESTORE VECTOR 114
                                        ;EXIT   TST
                                        ;
                                        ;TAG PARITY ABORT ROUTINE:
                                        ;                               ILLEGAL PARITY INTERRUPT
                                        ;                               RETURN
                                        ;ENDTST
                                        ;;*****************************************************************
     107360  000004                     TST14:  SCOPE
17310 107362  000240                             NOP
17311 107364  005737  003032                     TST     CCHPAS                  ;have done enough inclusive passes?
17312 107370  001002                             BNE     99$                     ; not yet
17313 107372  000240                             NOP                             ; debug aid
17314 107374  000453                             BR      TST15           ;;GO TO NEXT TEST
17315 107376  000240                     99$:    NOP
17316 107400  013737  000114  003012             MOV     @#114,  SLOC00          ;SAVE CONTENTS OF VECTOR 114
17317 107406  012737  107514  000114             MOV     #TAPABO,@#114           ;LET VECTOR POINT TO ABORT ROUTINE
17318 107414  042737  001000  177520             BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
17319 107422  005037  177744                     CLR     MSER                    ;CLEAR MSER
17320 107426  012737  002000  177746             MOV     #BIT10, CCR             ;SET WRITE WRONG TAG PARITY
17321 107434  005037  003162                     CLR     @#TSTLOC                ;WRITE LOCATION WITH BAD TAG PARITY
17322 107440  012737  000001  177746             MOV     #BIT00, CCR             ;CLEAR BIT 10 AND SET BIT 0
17323 107446  005737  003162                     TST     @#TSTLOC                ;READ TEST LOCATION
17324 107452  013737  177752  114150             MOV     HITMIS,RECDAT           ;SAVE HIT/MISS REGISTER
17325 107460  032737  000004  114150             BIT     #BIT02,RECDAT           ;IF BIT 1 SET IN HIT/MISS REGISTER
17326 107466  001401                             BEQ     2$                      ;THEN
17327 107470  104027                             ERROR   .27                     ;PARITY ERROR DON'T CAUSE A MISS
```

TEST - WRITE WRONG TAG PARITY

```
 17328 107472  013737  003012  000114  2$:     MOV     SLOC00, @#114           ;RESTORE CONTENTS OF VECTOR 114
 17329 107500  005037  177744                  CLR     MSER                    ;CLEAR ERRORS
 17330 107504  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
 17331 107512  000404                          BR      TST15           ;;GO TO NEXT TEST
 17332
 17333
 17334 107514  011637  001122          TAPABO: MOV     (SP),$BDADR             ;SAVE ADDRESS THAT CAUSE ABORT
 17335 107520  104007                          ERROR   +7                      ;ILLEGAL PARITY INTERRUPT
 17336 107522  000002                          RTI
 17337
```

```
17339                                   .SBTTL  TEST  - PARITY ABORT TEST
17340                                   ;PARITY ABORT TEST - THIS TEST WILL VERIFY THAT WHEN CCR<7,0> =
17341                                   ;1,0, AN ABORT OCCURS ON THE EXECUTION OF AN INSTRUCTION THAT HAS
17342                                   ;BEEN WRITTEN WITH WRONG PARITY.
17343                                   ;
17344                                   ;
17345                                   ;BGNTST
17346                                   ;SAVE VECTOR 114
17347                                   ;SAVE VECTOR 4
17348                                   ;LET VECTOR 114 POINT TO ABORT ROUTINE
17349                                   ;LET VECTOR 4 POINT TO ERROR ROUTINE
17350                                   ;CLEAR EXPECTING ABORT FLAG
17351                                   ;SET CCR<10> WRITE WRONG TAG PARITY BIT
17352                                   ;WRITE TEST ADDRESS
17353                                   ;CLEAR CCR<10>
17354                                   ;SET CCR TO #200 ENABLE PARITY ABORTS
17355                                   ;SET EXPECTING ABORT FLAG
17356                                   ;READ TEST ADDRESS
17357                                   ;IF ABORT FLAG NE 0 THEN
17358                                   ;.      ERROR IN PARITY ABORT LOGIC
17359                                   ;ENDIF
17360                                   ;RESTORE VECTOR 114
17361                                   ;RESTORE VECTOR 4
17362                                   ;EXIT   TST
17363                                   ;
17364                                   ;ABORT ROUTINE: IF EXPECTING ABORT FLAG NOT SET THEN
17365                                   ;                .      ERROR NO ABORT SHOULD HAVE OCCURRED
17366                                   ;                ELSE
17367                                   ;                .      CLEAR (EXPECTING) ABORT FLAG
17368                                   ;                ENDIF
17369                                   ;                IF MSER NOT EQUAL TO 100040 THEN
17370                                   ;                .      PARITY ABORT LOGIC DOESN'T SET MSER PROPERLY
17371                                   ;                ENDIF
17372                                   ;                IF PC = UPDATED PC THEN
17373                                   ;                .      ILLEGAL PARITY ABORT
17374                                   ;                ENDIF
17375                                   ;                RETURN
17376                                   ;
17377                                   ;ENDTST
17378                                   ;;*****************************************************************
     107524  000004                   TST15:  SCOPE
17379 107526  000240                           NOP
17380 107530  005737  003032                   TST     CCHPAS                  ;have done enough inclusive passes?
17381 107534  001002                           BNE     99$                     ; not yet
17382 107536  000240                           NOP                             ; debug aid
17383 107540  000504                           BR      TST16           ;;GO TO NEXT TEST
17384 107542  000240                   99$:    NOP
17385
17386 107544  013737  000114  003012         MOV     @#114,  SLOC00          ;SAVE VECTOR 114
17387 107552  013737  000004  003014         MOV     @#4,    SLOC01          ;SAVE VECTOR 4
17388 107560  012737  107676  000114         MOV     #ABORTR,@#114           ;LET VECTOR 114 POINT TO ABORT ROUTINE
17389                                   ;
17390                                   ; TO AVOID CONFUSION ALLOCATE IN CACHE FOLLOWING INSTRUCIONS
17391                                   ;
17392 107566  012704  107566                   MOV     #.,R4                   ;START WITH CURRENT
17393 107572  005724                   1$:     TST     (R4)+                   ;READ A WORD
17394 107574  022704  107676                   CMP     #ABORTR,R4              ;GOT TO ABORT ROUTINE?
```

```
17395 107600  001374                          BNE     1$                      ;IF NOT, KEEP ON ALLOCATING
17396 107602  005000                          CLR     R0                      ;CLEAR EXPECTING ABORT FLAG
17397 107604  042737  001000  177520          BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
17398 107612  012737  002000  177746          MOV     #BIT10, CCR             ;SET WRITE WRONG PARITY BIT
17399 107620  005037  003162                  CLR     @#TSTLOC                ;WRITE TEST LOCATION WITH BAD PARITY
17400 107624  012737  000200  177746          MOV     #BIT07, CCR             ;ENABLE ABORTS, CLEAR WWTP BIT
17401 107632  005100                          COM     R0                      ;SET EXPECTING ABORT FLAG
17402 107634  005737  003162          ABORTI: TST     @#TSTLOC                ;READ TEST LOCATION (SHOULD CAUSE ABORT)
17403 107640  005700                          TST     R0                      ;IF ABORT FLAG NOT EQUAL ZERO
17404 107642  001401                          BEQ     1$                      ;THEN
17405 107644  104034                          ERROR   .34                     ;PARITY ABORT LOGIC DOESN'T WORK
17406 107646  052737  001000  177520  1$:     BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
17407 107654  013737  003012  000114          MOV     SLOC00, @#114           ;RESTORE VECTOR 114
17408 107662  013737  003014  000004          MOV     SLOC01, @#4             ;RESTORE VECTORE 4
17409 107670  005037  177744                  CLR     MSER
17410 107674  000426                          BR      TST16           ;;GO TO NEXT TEST
17411
17412
17413 107676  013703  177744          ABORTR: MOV     MSER,R3                 ;SAVE MSER
17414 107702  005700                          TST     R0                      ;IF EXPECTING ABORT FLAG NOT SET
17415 107704  001004                          BNE     1$                      ;THEN
17416 107706  011637  001122                  MOV     (SP),   $BDADR          ;SAVE ABORT ADDRESS
17417 107712  104007                          ERROR   .7                      ;ILLEGAL PARITY INTERRUPT
17418 107714  000401                          BR      2$                      ;ELSE
17419 107716  005000                  1$:     CLR     R0                      ;CLEAR (EXPECTING) ABORT FLAG
17420 107720  022737  100040  177744  2$:     CMP     #100040,MSER            ;IF MSER NOT EQUAL TO 100040
17421 107726  001404                          BEQ     3$                      ;THEN
17422 107730  012737  100040  001124          MOV     #100040,$GDDAT          ;SAVE PROPER MSER SETTING
17423 107736  104035                          ERROR   .35                     ;PARITY ABORT DON'T SET MSER PROPERLY
17424 107740  021627  107640          3$:     CMP     (SP),   #ABORTI+4       ;IF PC EQUAL TO UPDATE PC
17425 107744  001401                          BEQ     4$                      ;THEN
17426 107746  104007                          ERROR   .7                      ;ILLEGAL PARITY INTERRUPT
17427 107750  000002                  4$:     RTI                             ;RETURN
17428
```

```
                                .SBTTL  TEST  - PARITY INTERRUPT TEST
                                ;PARITY INTERRUPT TEST - THIS TEST WILL VERIFY THAT WHEN CCR<7,0> =
                                ;0,0, A PARITY INTERRRUPT OCCURS AFTER EXECUTION OF AN INSTRUCTION
                                ;THAT HAS BEEN WRITTEN WITH WRONG PARITY.
                                ;
                                ;BGNTST
                                ;SAVE CONTENTS OF 114
                                ;SETUP VECTOR 114 TO POINT TO INTERRUPT ROUTINE
                                ;CLEAR EXPECTING INTERRUPT FLAG
                                ;WRITE TEST ADDRESS WITH BAD PARITY
                                ;SET EXPECTING INTERRUPT FLAG
                                ;READ TEST ADDRESS
                                ;IF INTERRUPT FLAG NE 0 THEN
                                ;.      PARITY INTERRUPT LOGIC DOESN'T WORK
                                ;ENDIF
                                ;RESTORE CONTENTS OF VECTOR 114
                                ;EXIT   TST
                                ;
                                ;INTERRUPT ROUTINE:     IF EXPECTING INTERRUPT FLAG NE 1 THEN
                                ;                       .       ERROR NO INTERRUPT SHOULD HAVE OCCURRED
                                ;                       ELSE
                                ;                       .       CLEAR (EXPECTING) INTERRUPT FLAG
                                ;                       ENDIF
                                ;                       IF SAVED PC NE TO UPDATED PC THEN
                                ;                       .       IF PC = TEST INSTRUCTION PC THEN
                                ;                       .       .       ERROR INSTRUCTION ABORTED
                                ;                       .       ELSE
                                ;                       .       .       ILLEGAL PARITY ABORT
                                ;                       .       ENDIF
                                ;                       ENDIF
                                ;                       IF MSER NE #340 THEN
                                ;                       .       PARITY INTERRUPT DOESN'T SET MSER PROPERLY
                                ;                       ENDIF
                                ;                       RETURN
                                ;
                                ;ENDTST
                                ;;*****************************************************************
    107752  000004              TST16:  SCOPE
000240      107754  000240              NOP
005737      107756  005737  003032      TST     CCHPAS                  ;have done enough inclusive passes?
001002      107762  001002              BNE     99$                     ; not yet
000240      107764  000240              NOP                             ; debug aid
000473      107766  000473              BR      TST17           ;;GO TO NEXT TEST
000240      107770  000240      99$:    NOP
                    107772  013737  000114  003012  MOV     @#114,  SLOC00          ;SAVE CONTENTS OF VECTOR 114
                    110000  012737  110106  000114  MOV     #INTERR,@#114           ;LET VECTOR POINT TO INTERRUPT ROUTINE
                                ;
                                ; TO AVOID CONFUSION ALLOCATE IN CACHE FOLLOWING INSTRUCIONS
                                ;
                    110006  012704  110006          MOV     #.,R4                   ;START WITH CURRENT
                    110012  005724          1$:     TST     (R4)+                   ;READ A WORD
                    110014  022704  110106          CMP     #INTERR,R4              ;GOT TO INTERRUPT ROUTINE?
                    110020  001374                  BNE     1$                      ;IF NOT, KEEP ON ALLOCATING
                    110022  005001                  CLR     R1                      ;CLEAR EXPECTING INTERRUPT FLAG
                    110024  042737  001000  177520  BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
                    110032  052737  000100  177746  BIS     #BIT06, CCR             ;SET WRITE WRONG DATA PARITY
```

TEST - PARITY INTERRUPT TEST

```
 17486 110040  005037  003162                CLR     @#TSTLOC                ;WRITE LOCATION WITH BAD DATA PARITY
 17487 110044  005037  177746                CLR     CCR                     ;CLEAR WRITE WRONG DATA PARITY
 17488 110050  005101                       COM     R1                      ;SET EXPECTING INTERRUPT FLAG
 17489 110052  005737  003162        INTRPC: TST     @#TSTLOC                ;READ TEST LOCATION
 17490 110056  005701                       TST     R1                      ;IF INTERRUPT FLAG NOT EQUAL ZERO
 17491 110060  001401                       BEQ     1$                      ;THEN
 17492 110062  104036                       ERROR   +36                     ;PARITY INTERRUPT LOGIC DOESN'T WORK
 17493 110064  052737  001000  177520  1$:  BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
 17494 110072  013737  003012  000114        MOV     SLOC00, @#114           ;RESTORE VECTOR 114
 17495 110100  005037  177744                CLR     MSER                    ;CLEAR MSER
 17496 110104  000424                       BR      TST17           ;;GO TO NEXT TEST
 17497
 17498
 17499 110106  005701        INTERR: TST     R1                      ;IF EXPECTING INTERRUPT FLAG NOT SET
 17500 110110  001004                       BNE     1$                      ;THEN
 17501 110112  011637  001122                MOV     (SP),   $BDADR          ;SAVE INTERRUPT ADDRESS
 17502 110116  104007                       ERROR   +7                      ;ILLEGAL PARITY INTERRUPT
 17503 110120  000401                       BR      2$                      ;ELSE
 17504 110122  005001                1$:    CLR     R1                      ;CLEAR (EXPECTING) INTERRUPT FLAG
 17505 110124  021627  110056        2$:    CMP     (SP),   #INTRPC+4       ;IF SAVED PC NOT EQUAL TO UPDATED PC
 17506 110130  001401                       BEQ     4$                      ;THEN
 17507 110132  104007                       ERROR   +7                      ;ILLEGAL PARITY INTERRUPT
 17508 110134  022737  000340  177744  4$:  CMP     #340,   MSER            ;IF MSER NOT EQUAL TO EXPECTED VALUE
 17509 110142  001404                       BEQ     5$                      ;THEN
 17510 110144  012737  000340  001124        MOV     #340,   $GDDAT          ;SAVE PROPER MSER
 17511 110152  104035                       ERROR   +35                     ;PARITY INTERRUPT DON'T SET MSER PROPERLY
 17512 110154  000002                5$:    RTI                             ;RETURN
 17513
```

```
 17515                                  .SBTTL  TEST    - MISCELLANEOUS PARITY TEST
 17516                                  ;MISCELLANEOUS PARITY TEST - THIS TEST CHECKS THAT BYPASS CYCLES WITH
 17517                                  ;PARITY ERRORS CAUSE CACHE HIT RESPONSE AND THAT FORCE MISS CYCLES
 17518                                  ;IGNORE PARITY ERRORS.
 17519                                  ;
 17520                                  ;BGNTST
 17521                                  ;WRITE A LOCATION WITH BAD PARITY
 17522                                  ;SET BYPASS IN CCR
 17523                                  ;READ THE LOCATION BACK
 17524                                  ;IF NO HIT OR MSER NOT SET THEN
 17525                                  ;.      ERROR
 17526                                  ;ENDIF
 17527                                  ;WRITE A LOCATION WITH BAD PARITY AND SET BYPASS
 17528                                  ;IF MSER SET WHILE READING IT BACK
 17529                                  ;.      ERROR
 17530                                  ;ENDIF
 17531                                  ;ENDTST
 17532                                  ;;*****************************************************************
     110156  000004                     TST17:  SCOPE
 17533 110160  000240                             NOP
 17534 110162  005737  003032                     TST     CCHPAS                  ;have done enough inclusive passes?
 17535 110166  001002                             BNE     99$                     ; not yet
 17536 110170  000240                             NOP                             ; debug aid
 17537 110172  000503                             BR      TST20           ;;GO TO NEXT TEST
 17538 110174  000240                     99$:    NOP
 17539
 17540 110176  012703  110176                     MOV     #.,R3                           ;START WITH CURRENT INSTRUCTION
 17541 110202  005723                     10$:    TST     (R3)+                   ;READ A WORD
 17542 110204  022703  110354                     CMP     #4$,R3                  ;LAST WORD?
 17543 110210  001374                             BNE     10$                     ;IF NOT, CONTINUE
 17544                                  ;
 17545                                  ; CHECK BYPASS AND BAD PARITY
 17546                                  ;
 17547 110212  013737  000114  003012             MOV     @#114,  SLOC00          ;SAVE PARITY VECTOR
 17548 110220  012737  110264  000114             MOV     #1$,@#114               ;POINT NEW VECTOR
 17549 110226  042737  001000  177520             BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
 17550 110234  052737  002101  177746             BIS     #BIT10!BIT06!BIT00,CCR  ;DATA AND TAG PAR., NO INT.
 17551 110242  005037  003162                     CLR     @#TSTLOC                ;WRITE CYCLE
 17552 110246  012737  001000  177746             MOV     #BIT09,CCR              ;SET BYPASS
 17553 110254  005737  003162                     TST     @#TSTLOC                ;BYPASS WITH WRONG PARITY
 17554 110260  104045                             ERROR   +45                     ;ERROR
 17555 110262  000407                             BR      2$                      ;
 17556 110264  062706  000004             1$:     ADD     #4,SP                   ;ADJUST STACK
 17557 110270  032737  000340  177744             BIT     #340,MSER               ;MSER OK?
 17558 110276  001001                             BNE     2$                      ;IF YES, BRANCH
 17559 110300  104042                             ERROR   +42                     ;BYPASS WRONG
 17560                                  ;
 17561                                  ; CHECK FORCE MISS AND BAD PARITY
 17562                                  ;
 17563 110302  005037  177744             2$:     CLR     MSER                    ;CLEAR MSER
 17564 110306  005037  177746                     CLR     CCR                     ;CLEAR CCR
 17565 110312  012737  110346  000114             MOV     #3$,@#114               ;POINT NEW VECTOR
 17566 110320  052737  002101  177746             BIS     #BIT10!BIT06!BIT00,CCR  ;DATA AND TAG PAR., NO INTER
 17567 110326  005037  003162                     CLR     @#TSTLOC                ;FORCE MISS WITH PARITY
 17568 110332  012737  000014  177746             MOV     #14,CCR                 ;FORCE MISS
 17569 110340  005737  003162                     TST     @#TSTLOC                ;ALLOCATE CACHE
 17570 110344  000403                             BR      4$
```

TEST - MISCELLANEOUS PARITY TEST

```
 17571 110346  104042                 3$:    ERROR   +42
 17572 110350  062706  000004                ADD     #4,SP                   ;ADJUST STACK
 17573 110354  005037  177746           4$:  CLR     CCR                     ;CLEAR CCR
 17574 110360  052737  001000  177520        BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
 17575 110366  013737  003012  000114        MOV     SLOC00,@#114            ;RESTORE PARITY VECTOR
 17576 110374  012737  000400  177746        MOV     #BIT08,CCR              ;FLUSH CACHE
 17577
```

```
17579                                   .SBTTL  TEST  - MEMORY SYSTEM ERROR REGISTER TEST
17580                                   ;MEMORY SYSTEM ERROR REGISTER TEST - THIS TEST WILL VERIFY THE
17581                                   ;FUNCTIONALITY OF BITS <15> AND <7:5> OF THE MEMORY SYSTEM ERROR
17582                                   ;REGISTER. THIS TEST WILL USE THE WRITE WRONG PARITY BITS (BITS<10>
17583                                   ;AND <6> OF THE CCR) TO WRITE BAD PARITY INTO A LOCATION. THE
17584                                   ;LOCATION WILL THEN BE READ WITH CACHE TRAPS ENABLED AND THE MSER
17585                                   ;WILL BE CHECKED AFTER THE ABORT FOR THE CORRECT BIT(S) BEING SET.
17586                                   ;THIS WILL BE DONE FOR ALL COMBINATIONS OF BITS<10> AND <6> FOR WORD
17587                                   ;ACCESSES THEN REPEATED FOR ALL COMBINATIONS OF BITS<10> AND <6> FOR
17588                                   ;BYTE ACCESSES. THE TEST WILL THEN BE REPEATED A THIRD TIME FOR BYTE
17589                                   ;ACCESSES AND ABORTS DISABLED. THE MSER SHOULD CONTAIN BITS <7:5>
17590                                   ;SET TO 1'S AND BIT <15> A ZERO FOR ALL COMBINATIONS.
17591                                   ;
17592                                   ;BGNTST
17593                                   ;SAVE CONTENTS OF VECTOR 114
17594                                   ;SETUP VECTOR TO POINT TO ABORT ROUTINE
17595                                   ;INITIALIZE LOOP COUNTER
17596                                   ;DO UNTIL ALL WORD COMBINATIONS CHECKED
17597                                   ;.      INITIALIZE MSER
17598                                   ;.      SETUP CCR FROM CCR TABLE
17599                                   ;.      READ TEST LOCATION TO ALLOCATE LOCATION WITH DESIRED PARITY
17600                                   ;.      CLEAR <10,6> FROM CCR
17601                                   ;.      SETUP CCR FOR ABORT
17602                                   ;.      READ TEST LOCATION                      ;THIS COULD CAUSE TRAP
17603                                   ;.      IF EXPECTED DATA NE RECEIVED DATA THEN
17604                                   ;.      .       ERROR IN SETTING MSER
17605                                   ;.      ENDIF
17606                                   ;.      UPDATE LOOP COUNTER
17607                                   ;ENDDO
17608                                   ;INITIALIZE LOOP COUNTER
17609                                   ;DO UNTIL ALL BYTE COMBINATIONS CHECKED
17610                                   ;.      INITIALIZE MSER
17611                                   ;.      SETUP CCR FROM CCR TABLE
17612                                   ;.      READ TEST LOCATION TO ALLOCATE LOCATION WITH DESIRED PARITY
17613                                   ;.      CLEAR <10,6> FROM CCR
17614                                   ;.      SETUP CCR FOR ABORT
17615                                   ;.      READ LOW BYTE TEST LOCATION
17616                                   ;.      GET EXPECTED BYTE DATA FROM TABLE
17617                                   ;.      IF EXPECTED DATA NE RECEIVED DATA THEN
17618                                   ;.      .       ERROR IN SETTING MSER
17619                                   ;.      ENDIF
17620                                   ;.      INITIALIZE MSER
17621                                   ;.      READ HIGH BYTE TEST LOCATION
17622                                   ;.      GET EXPECTED BYTE DATA FROM TABLE
17623                                   ;.      IF EXPECTED DATA NE RECEIVED DATA THEN
17624                                   ;.      .       ERROR IN SETTING MSER
17625                                   ;.      ENDIF
17626                                   ;.      INCREMENT LOOP COUNTER
17627                                   ;ENDDO
17628                                   ;INITIALIZE LOOP COUNTER
17629                                   ;DO UNTIL ALL BYTE COMBINATIONS CHECKED
17630                                   ;.      INITIALIZE MSER
17631                                   ;.      SETUP CCR FROM CCR TABLE
17632                                   ;.      READ TEST LOCATION TO ALLOCATE LOCATION WITH DESIRED PARITY
17633                                   ;.      CLEAR <10,6> FROM CCR
17634                                   ;.      SETUP CCR FOR NO ABORTS
17635                                   ;.      READ LOW BYTE TEST LOCATION
```

```
                                ;.      IF RECEIVED DATA NE TO #340 THEN
                                ;.      .       ERROR IN SETTING MSER
                                ;.      ENDIF
                                ;.      INITIALIZE MSER
                                ;.      READ HIGH BYTE TEST LOCATION
                                ;.      IF RECEIVED DATA NE #340 THEN
                                ;.      .       ERROR IN SETTING MSER
                                ;.      ENDIF
                                ;.      INCREMENT LOOP COUNTER
                                ;ENDDO
                                ;EXIT   TST
                                ;
                                ;CCR TABLE:     0
                                ;               100
                                ;               2000
                                ;               2100
                                ;EXPECTED WORD DATA:    0
                                ;                       100300
                                ;                       100040
                                ;                       100340
                                ;EXPECTED BYTE DATA:    0
                                ;                       0
                                ;                       100100
                                ;                       100200
                                ;                       100040
                                ;                       100040
                                ;                       100140
                                ;                       100240
                                ;ABORT ROUTINE: RTI
                                ;
                                ;ENDTST
                                ;;*****************************************************************
    110402  000004              TST20:  SCOPE
17668 110404  000240                    NOP
17669 110406  005737  003032            TST     CCHPAS          ;have done enough inclusive passes?
17670 110412  001003                    BNE     99$             ; not yet
17671 110414  000240                    NOP                     ; debug aid
17672 110416  000137  111104            JMP     10$             ; yes skip this
17673 110422  000240              99$:  NOP
17674
17675 110424  013737  000114  003012    MOV     @#114,  SLOC00  ;SAVE CONTENTS OF VECTOR 114
17676 110432  012737  111146  000114    MOV     #ABROUT,@#114   ;SETUP VECTOR TO POINT TO ABORT ROUTINE
17677 110440  012704  000004            MOV     #4,     R4      ;INITIALIZE LOOP COUNTER
17678 110444  012700  111106            MOV     #CCRTBL,R0      ;GET ADDRESS OF CCR TABLE
17679 110450  012701  111116            MOV     #EXPWDT,R1      ;GET ADDRESS OF EXPECTED DATA TABLE
17680 110454  042737  001000  177520    BIC     #1000,BCSR      ;DISABLE HALT ON BREAK
17681 110462  005037  177744      1$:   CLR     MSER            ;INITIALIZE MSER DATA
17682 110466  012037  177746            MOV     (R0)+,  CCR     ;SETUP CCR FROM CCR TABLE
17683 110472  005037  003162            CLR     @#TSTLOC        ;ALLOCATE CACHE LOC. WITH DESIRED PARITY
17684 110476  012737  000200  177746    MOV     #BIT07, CCR     ;SETUP CCR TO ABORT POSSIBLE BAD PARITY
17685 110504  005737  003162            TST     @#TSTLOC        ;READ TEST LOCATION
17686 110510  021137  177744            CMP     (R1),   MSER    ;IF RECEIVED DATA NOT EQUAL TO EXPECTED
17687 110514  001403                    BEQ     2$              ;DATA THEN
17688 110516  011137  001124            MOV     (R1),   $GDDAT  ;SAVE PROPER MSER SETTING
17689 110522  104035                    ERROR   +35             ;MSER NOT SET PROPERLY
17690 110524  005721              2$:   TST     (R1)+           ;INCREMENT POINTER THRU MSER TABLE
17691 110526  005737  023162            TST     @#TSTLOC+8192.  ;TO INSURE MISS ON THE NEXT LOOP
```

```
17692 110532  077425                        SOB     R4,     1$              ;LOOP UNTIL ALL COMBINATIONS CHECKED
17693 110534  052737  001000  177520        BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
17694
17695                               ;CHECK BYTE OPERATIONS NOW
17696 110542  012704  000004                MOV     #4,     R4              ;INITIALIZE LOOP COUNTER
17697 110546  012700  111106                MOV     #CCRTBL,R0              ;GET ADDRESS OF CCR TABLE
17698 110552  012701  111126                MOV     #EXPBDT,R1              ;GET ADDRESS OF EXPECTED BYTE DATA TABLE
17699 110556  042737  001000  177520        BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
17700 110564  005037  177744          3$:   CLR     MSER                    ;INITIALIZE MSER
17701 110570  005737  003162                TST     @#TSTLOC                ;ALLOCATE TO HAVE WRITE BYTE HIT
17702 110574  011037  177746                MOV     (R0),   CCR             ;SETUP CCR FROM TABLE
17703 110600  105037  003162                CLRB    @#TSTLOC                ;ALLOCATE CACHE LOC. WITH DESIRED PARITY
17704 110604  012737  000200  177746        MOV     #BIT07, CCR             ;SETUP CCR TO ABORT POSSIBLE BAD PARITY
17705 110612  105737  003162                TSTB    @#TSTLOC                ;READ LOW BYTE OF TEST LOCATION
17706 110616  021137  177744                CMP     (R1),   MSER            ;IF RECEIVED DATA NOT EQUAL TO EXPECTED
17707 110622  001403                        BEQ     4$                      ;THEN
17708 110624  011137  001124                MOV     (R1),   $GDDAT          ;SAVE PROPER MSER SETTING
17709 110630  104035                        ERROR   +35                     ;MSER NOT SET PROPERLY
17710 110632  005721                  4$:   TST     (R1)+                   ;INCREMENT POINTER THRU MSER TABLE
17711 110634  005037  177744                CLR     MSER                    ;INITIALIZE MSER
17712 110640  005737  003162                TST     @#TSTLOC                ;ALLOCATE TO HAVE WRITE BYTE HIT
17713 110644  012037  177746                MOV     (R0)+,  CCR             ;SETUP CCR FROM TABLE
17714 110650  105037  003163                CLRB    @#TSTLOC+1              ;ALLOCATE CACHE LOC. WITH DESIRED PARITY
17715 110654  012737  000200  177746        MOV     #BIT07, CCR             ;SETUP CCR TO ABORT POSSIBLE BAD PARITY
17716 110662  105737  003163                TSTB    @#TSTLOC+1              ;READ HIGH BYTE OF TEST LOCATION
17717 110666  021137  177744                CMP     (R1),   MSER            ;IF RECEIVED DATA NOT EQUAL TO EXPECTED
17718 110672  001403                        BEQ     5$                      ;THEN
17719 110674  011137  001124                MOV     (R1),   $GDDAT          ;SAVE PROPER MSER SETTING
17720 110700  104035                        ERROR   +35                     ;MSER NOT SET PROPERLY
17721 110702  005721                  5$:   TST     (R1)+                   ;INCREMENT POINTER THRU MSER TABLE
17722 110704  077451                        SOB     R4,     3$              ;DO UNTIL ALL BYTE COMBINATIONS CHECKED
17723 110706  052737  001000  177520        BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
17724                               ;REPEAT WITHOUT ABORT
17725 110714  012704  000003                MOV     #3,     R4              ;INITIALIZE LOOP COUNTER
17726 110720  012700  111110                MOV     #CCRTBL+2,R0            ;GET ADDRESS OF CCR TABLE
17727 110724  042737  001000  177520        BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
17728 110732  005037  177744          6$:   CLR     MSER                    ;INITIALIZE MSER
17729 110736  005737  003162                TST     @#TSTLOC                ;ALLOCATE CACHE LOC.
17730 110742  011037  177746                MOV     (R0),   CCR             ;SETUP CCR FROM TABLE
17731 110746  105037  003162                CLRB    @#TSTLOC                ;ALLOCATE CACHE LOC. WITH DESIRED PARITY
17732 110752  012737  000001  177746        MOV     #BIT00, CCR             ;SETUP CCR TO NOT ABORT
17733 110760  105737  003162                TSTB    @#TSTLOC                ;READ LOW BYTE OF TEST LOCATION
17734 110764  022737  000340  177744        CMP     #340,   MSER            ;IF RECEIVED DATA NOT EQUAL TO EXPECTED
17735 110772  001404                        BEQ     7$                      ;THEN
17736 110774  012737  000340  001124        MOV     #340,   $GDDAT          ;SAVE PROPER MSER SETTING
17737 111002  104035                        ERROR   +35                     ;MSER NOT SET PROPERLY
17738 111004  005037  177744          7$:   CLR     MSER                    ;INITIALIZE MSER
17739 111010  005737  003162                TST     @#TSTLOC                ;ALLOCATE CACHE LOC.
17740 111014  012037  177746                MOV     (R0)+,  CCR             ;SETUP CCR FROM TABLE
17741 111020  105037  003163                CLRB    @#TSTLOC+1              ;ALLOCATE CACHE LOC. WITH DESIRED PARITY
17742 111024  012737  000001  177746        MOV     #BIT00, CCR             ;SETUP CCR TO NOT ABORT
17743 111032  105737  003163                TSTB    @#TSTLOC+1              ;READ HIGH BYTE OF TEST LOCATION
17744 111036  022737  000340  177744        CMP     #340,   MSER            ;IF RECEIVED DATA NOT EQUAL TO EXPECTED
17745 111044  001404                        BEQ     8$                      ;THEN
17746 111046  012737  000340  001124        MOV     #340,   $GDDAT          ;SAVE PROPER MSER SETTING
17747 111054  104035                        ERROR   +35                     ;MSER NOT SET PROPERLY
17748 111056  077453                  8$:   SOB     R4,     6$              ;DO UNTIL ALL BYTE COMBINATIONS CHECKED
```

TEST - MEMORY SYSTEM ERROR REGISTER TEST

```
 17749 111060  005037  177744                CLR     MSER                    ;CLEAR ERROR REGISTER
 17750 111064  052737  001000  177520        BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
 17751 111072  013737  003012  000114        MOV     SLOC00, @#114           ;RESTORE VECTOR 114
 17752 111100  005037  177746                CLR     CCR                     ;CLEAR ALL BIT IN CCR
 17753 111104                        10$:
       111104  000421                        BR      TST21           ;;GO TO NEXT TEST
 17754
 17755
 17756 111106  000000                CCRTBL: .WORD   0
 17757 111110  000100                        .WORD   100
 17758 111112  002000                        .WORD   2000
 17759 111114  002100                        .WORD   2100
 17760
 17761 111116  000000                EXPWDT: .WORD   0
 17762 111120  100300                        .WORD   100300
 17763 111122  100040                        .WORD   100040
 17764 111124  100340                        .WORD   100340
 17765
 17766 111126  000000                EXPBDT: .WORD   0
 17767 111130  000000                        .WORD   0
 17768 111132  100100                        .WORD   100100
 17769 111134  100200                        .WORD   100200
 17770 111136  100040                        .WORD   100040
 17771 111140  100040                        .WORD   100040
 17772 111142  100140                        .WORD   100140
 17773 111144  100240                        .WORD   100240
 17774
 17775 111146  000002                ABROUT: RTI
 17776
 17777
```

```
17779                                   .SBTTL  TEST  - CHECK PARITY ABORTS BLOCKED BY NON-EXISTENT MEMORY ABORT
17780                                   ;CHECK PARITY ABORTS BLOCKED BY NON-EXISTENT MEMORY ABORT - THIS TEST WILL
17781                                   ;VERIFY THAT IF A PARITY ERROR OCCURS ON THE SAME ADDRESS REFERENCE AS A
17782                                   ;NON-EXISTENT MEMORY ERROR THAT THE CACHE DATA PATH GATE ARRAY BLOCKS THE
17783                                   ;PARITY ERROR TO THE J-11 CHIP SET. THIS WILL BE DONE BY USING THE DIAGNOSTIC
17784                                   ;BIT TO CAUSE A PARITY ERROR IN A CACHE REFERENCE THAT DOES NOT HAVE A
17785                                   ;CORRESPONDING ADDRESS IN MAIN MEMORY. THE ADDRESS WILL THEN BE READ CAUSING
17786                                   ;BOTH A CACHE PARITY ERROR AND A NON-EXISTENT MEMORY ERROR. TO AVOID HAVING
17787                                   ;TO SIZE THE ENTIRE MEMORY TO FIND A NON-EXISTENT MEMORY ADDRESS THIS TEST
17788                                   ;WILL USE THE LARGEST NON-I/O ADDRESS (17757776). THE TEST WILL FIRST READ
17789                                   ;THIS ADDRESS AND IF A NXM TRAP OCCURS THE TEST WILL BE DONE. IF THE ACCESS
17790                                   ;TO THIS ADDRESS DOES NOT TRAP THEN THE TEST WILL BE SKIPPED.
17791                                   ;
17792                                   ;BGNTST
17793                                   ;SAVE CONTENTS OF VECTOR 4
17794                                   ;SAVE CONTENTS OF VECTOR 114
17795                                   ;LET VECTOR 4 POINT TO CONTINUE TESTING (A:)
17796                                   ;LET PAR6 = #177400
17797                                   ;ACCESS ADDRESS 157776 (PHYSICAL 17757776)
17798                                   ;IF NO TRAP THEN
17799                                   ;.      GOTO ENDTST
17800                                   ;ENDIF
17801                                   ;A:
17802                                   ;SET DIAGNOSTIC AND WRITE WRONG PARITY BITS IN CCR
17803                                   ;WRITE ADDRESS 157776
17804                                   ;CLEAR CCR
17805                                   ;SET PARITY ERROR ABORT BIT IN CCR
17806                                   ;LET VECTOR 4 POINT TO CONTINUE TESTING (B:)
17807                                   ;LET VECTOR 114 POINT TO NXM-PARITY ERROR ROUTINE
17808                                   ;READ ADDRESS 157776
17809                                   ;IF NO TRAP THEN
17810                                   ;.      ERROR IN ABORT LOGIC
17811                                   ;ENDIF
17812                                   ;B:
17813                                   ;CLEAR CCR
17814                                   ;RESTORE CONTENTS OF VECTOR 114
17815                                   ;RESTORE CONTENTS OF VECTOR 4
17816                                   ;EXIT   TST
17817                                   ;
17818                                   ;
17819                                   ;NXM-PARITY ERROR ROUTINE:      RESET STACK AFTER TRAP
17820                                   ;                                       PARITY ABORT NOT BLOCKED BY NXM
17821                                   ;                               GOTO B:
17822                                   ;ENDTST
17823                                   ;;*****************************************************************
    111150  000004                      TST21:  SCOPE
17824 111152  000240                              NOP
17825 111154  005737  003032                      TST     CCHPAS                  ;have done enough inclusive passes?
17826 111160  001002                              BNE     99$                     ; not yet
17827 111162  000240                              NOP                             ; debug aid
17828 111164  000475                              BR      TST22           ;;GO TO NEXT TEST
17829 111166  000240                      99$:    NOP
17830
17831 111170  013737  000004  003012              MOV     @#4,    SLOC00          ;SAVE CONTENTS OF VECTOR 4
17832 111176  013737  000114  003014              MOV     @#114,  SLOC01          ;SAVE CONTENTS OF VECTOR 114
17833 111204  012737  111232  000004              MOV     #1$,    @#4             ;LET VECTOR 4 POINT TO CONTINUE TESTING
17834 111212  012737  177400  172354              MOV     #177400,$IPAR6          ;LET PAR6 = OFFSET TO HIGHEST MEMORY
```

TEST - CHECK PARITY ABORTS BLOCKED BY NON-EXISTENT MEMORY ABO

```
 17835 111220  005237  177572                INC     SR0                     ;TURN ON MMU
 17836 111224  005737  157776                TST     @#157776                ;ACCESS ADDRESS 17757776
 17837 111230  000431                        BR      ABOEXT                  ;IF NO TRAP SKIP TEST
 17838 111232  062706  000004          1$:   ADD     #4,     SP              ;RESET STACK AFTER TRAP
 17839 111236  042737  001000  177520        BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
 17840 111244  012737  000102  177746        MOV     #102,   CCR             ;SET DIAG. AND WRITE WRONG PARITY BITS
 17841 111252  005037  157776                CLR     @#157776                ;WRITE TO ADDRESS 17757776
 17842 111256  012737  000200  177746        MOV     #200,   CCR             ;CLEAR CCR AND SET PARITY ABORT BIT
 17843 111264  012737  111310  000004        MOV     #2$,    @#4             ;LET VECTOR 4 POINT TO CONTINUE TESTING
 17844 111272  012737  111350  000114        MOV     #NXMPAR,@#114           ;LET VECTOR 114 POINT TO ERROR ROUTINE
 17845 111300  005737  157776                TST     @#157776                ;READ ADDRESS 17757776 (SHOULD TRAP)
 17846 111304  104037                        ERROR   +37                     ;NXM AND PARITY ABORT DIN'T HAPPEN
 17847 111306  000402                        BR      ABOEXT                  ;GO EXIT TEST
 17848 111310  062706  000004          2$:   ADD     #4,     SP              ;RESET STACK AFTER TRAP
 17849 111314  005037  177746    ABOEXT:   CLR     CCR                     ;CLEAR CCR FOR EXIT
 17850 111320  005037  177572              CLR     SR0                     ;DISABLE MMU
 17851 111324  052737  001000  177520        BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
 17852 111332  013737  003012  000004        MOV     SLOC00, @#4             ;RESTORE VECTOR 4
 17853 111340  013737  003014  000114        MOV     SLOC01, @#114           ;RESTORE VECTOR 114
 17854 111346  000404                        BR      TST22           ;;GO TO NEXT TEST
 17855
 17856
 17857
 17858 111350  062706  000004    NXMPAR:   ADD     #4,     SP              ;RESET STACK AFTER TRAP
 17859 111354  104040                      ERROR   +40                     ;PARITY ABORT NOT BLOCKED BY NXM TRAP
 17860 111356  000002                      RTI
 17861
```

```
                                .SBTTL  TEST  - MULTIPROCCESSING INSTRUCTION TESTS
                                ;MULTIPROCCESSING INSTRUCTION TESTS - THIS TEST WILL VERIFY THAT THE MULTI-
                                ;PROCCESSING INSTRUCTIONS DO A BYPASS OF THE CACHE. THIS TEST WILL NOT VERIFY
                                ;THE REST OF THE FUNCTIONALITY OF THESE INSTRUCTIONS BECAUSE THAT WILL ALREADY
                                ;HAVE BEEN CHECKED IN THE BASE INSTRUCTION TESTS. THE TEST WILL FIRST ALLOCATE
                                ;AN ADDRESS IN CACHE THEN A TSTSET INSTRUCTION WILL BE DONE AT THAT ADDRESS.
                                ;A HIT SHOULD BE RECORDED ON THE ACCESS. NEXT, THE ADDRESS WILL BE READ AND
                                ;A MISS SHOULD BE RECORDED BECAUSE THE FORCED BYPASS ON THE TSTSET INSTRUCTION
                                ;SHOULD HAVE INVALIDATED THE CACHE ENTRY. THE SAME SEQUENCE WILL THEN BE
                                ;REPEATED FOR THE WRTLCK INSTRUCTION.
                                ;
                                ;BGNTST
                                ;READ TEST LOCATION TO ALLOCATE CACHE
                                ;DO TSTSET INSTRUCTION
                                ;IF HIT/MISS REGISTER BIT 3 NOT SET OR BIT 2 SET THEN
                                ;.       ERROR IN MULTI-PROCESSOR HOOKS
                                ;ENDIF
                                ;READ TEST LOCATION (ALSO ALLOCATES CACHE FOR WRTLCK)
                                ;DO WRTLCK INSTRUCTION
                                ;IF HIT/MISS REGISTER BIT 3 NOT SET OR BIT 2 SET THEN
                                ;.       ERROR IN MULTI-PROCESSOR HOOKS
                                ;ENDIF
                                ;READ TEST LOCATION
                                ;DO ASRB INSTRUCTION
                                ;IF HIT/MISS REGISTER BIT 3 NOT SET OR BIT 2 SET THEN
                                ;.       ERROR IN MULTI-PROCESSOR HOOKS
                                ;ENDIF
                                ;ENDTST
                                ;NOTE: THE CODE IS POSITION DEPENDENT

                                ;;*****************************************************************
 111360  000004                 TST22:  SCOPE
17894 111362  000240                    NOP
17895 111364  005737  003032              TST     CCHPAS                  ;have done enough inclusive passes?
17896 111370  001002                      BNE     99$                     ; not yet
17897 111372  000240                      NOP                             ; debug aid
17898 111374  000501                      BR      TST23           ;;GO TO NEXT TEST
17899 111376  000240              99$:    NOP
17900
17901 111400  042737  001000  177520      BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
17902 111406  005737  003162              TST     TSTLOC                  ;READ TEST LOCATION TO ALLOCATE CACHE
17903                             ;       TSTSET  TSTLOC                  ;DO TSTSET INSTRUCTION
17904 111412  007237              7$:     .WORD   7237                    ;THESE NEXT TWO LOCATIONS ARE THE TSTSET
17905 111414  003162                      .WORD   TSTLOC                  ;INSTR. BECAUSE THE ASSEMBLER WAS NOT READY
17906 111416  013737  177752  114150      MOV     HITMIS,RECDAT           ;STORE REGISTER
17907 111424  032737  000010  114150      BIT     #BIT3,RECDAT            ;IF HIT/MISS REGISTER BIT 3 NOT SET
17908 111432  001001                      BNE     1$                      ;THEN
17909 111434  104045                      ERROR   +45                     ;ERROR IN RECORDING HITS IN HIT/MISS
17910 111436  032737  000004  114150  1$: BIT     #BIT2,RECDAT            ;IF HIT/MISS REGISTER BIT 2 SET
17911 111444  001404                      BEQ     2$                      ;THEN
17912 111446  013737  111412  001126      MOV     @#7$,   $BDDAT          ;SAVE OPCODE
17913 111454  104041                      ERROR   +41                     ;MULTI-PROCESSOR HOOK INSTRUCTION DOESN'TT CAUSE MIS
S
17914
17915 111456  000240              ;2$:    WRTLCK  TSTLOC                  ;DO WRTLCK INSTRUCTION
17916 111460  000240              2$:     NOP                             ;PUT THE WRITE LOCK CODE ON THE RIGHT $$$
17917 111462  007337                      NOP                             ;BOUNDRY. $$$
17918 111464  003162                      .WORD   7337                    ;THESE NEXT TWO WORDS ARE THE WRTLCK
                                          .WORD   TSTLOC                  ;INSTR. BECAUSE THE ASSEMBLER WAS NOT READY
```

TEST - MULTIPROCCESSING INSTRUCTION TESTS

```
  17919 111466  013737  177752  114150          MOV     HITMIS,RECDAT           ;STORE REGISTER
  17920 111474  032737  000010  114150          BIT     #BIT3,RECDAT            ;IF HIT/MISS REGISTER BIT 3 NOT SET
  17921 111502  001001                          BNE     3$                      ;THEN
  17922 111504  104045                          ERROR   +45                     ;ERROR IN RECORDING HITS IN HIT/MISS
  17923 111506  032737  000004  114150  3$:     BIT     #BIT2,RECDAT            ;IF HIT/MISS REGISTER BIT 2 SET
  17924 111514  001404                          BEQ     4$                      ;THEN
  17925 111516  013737  111456  001126          MOV     @#2$,   $BDDAT          ;SAVE OPCODE
  17926 111524  104041                          ERROR   +41                     ;MULTI-PROCESSOR HOOK INSTRUCTION DOESN'TT CAUSE MIS
S
  17927 111526  106237  003162          4$:     ASRB    TSTLOC                  ;THE THIRD BYPASS INSTRUCTION
  17928 111532  013737  177752  114150          MOV     HITMIS,RECDAT           ;STORE REGISTER
  17929 111540  032737  000010  114150          BIT     #BIT3,RECDAT            ;IF HIT/MISS REGISTER BIT 3 NOT SET
  17930 111546  001001                          BNE     5$                      ;THEN
  17931 111550  104045                          ERROR   +45                     ;ERROR IN RECORDING HITS IN HIT/MISS
  17932 111552  052737  001000  177520  5$:     BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
  17933 111560  032737  000004  114150          BIT     #BIT2,RECDAT            ;IF HIT/MISS REGISTER BIT 2 SET
  17934 111566  001404                          BEQ     TST23           ;;EXIT TEST
  17935 111570  013737  111526  001126          MOV     @#4$,   $BDDAT          ;SAVE OPCODE
  17936 111576  104041                          ERROR   +41                     ;MULTI-PROCESSOR HOOK INSTRUCTION DOESN'TT CAUSE MIS
S
  17937
  17938
  17939
```

```
17941                                   .SBTTL  TEST  - DATA STORE RAM TESTS
17942                                   ;DATA STORE RAM TESTS - THERE ARE TWO TESTS FOR THE DATA STORE RAM.
17943                                   ;THE FIRST TEST WILL BE A NO DUAL ADDRESSING TEST AND THE SECOND
17944                                   ;TEST WILL BE A DATA RELIABILITY TEST. THE NO DUAL ADDRESSING TEST
17945                                   ;WILL FIRST WRITE EACH WORD OF THE CACHE RAM WITH ITS WORD ADDRESS
17946                                   ;AND VERIFY THE CONTENTS WITH A READ OF EACH ADDRESS. THEN EACH
17947                                   ;BYTE WILL BE WRITTEN WITH ITS ADDRESS AND CHECKED. THE DATA
17948                                   ;RELIABILITY TEST THAT WILL BE USED IS A TEST CALLED MOVING INVERSIONS.
17949                                   ;
17950                                   ;BGNTST 1
17951                                   ;SETUP MMU REGISTERS TO HAVE BYPASS ON KERNAL SPACE AND NO
17952                                   ;       BYPASS ON USER SPACE
17953                                   ;LET PS EQUAL KERNAL FOR CURRENT MODE AND USER FOR PREVIOUS
17954                                   ;       MODE
17955                                   ;ENABLE MMU
17956                                   ;GET FIRST ADDRESS OF 4K WORD DATA BUFFER
17957                                   ;CLEAR DATA TO BE WRITTEN
17958                                   ;SET DIAGNOSTIC BIT (BIT<1>) IN CCR
17959                                   ;DO UNTIL DATA TO BE WRITTEN EQUALS 20000(8)
17960                                   ;.      WRITE DATA TO ADDRESS
17961                                   ;.      ADD 2 TO ADDRESS
17962                                   ;.      ADD 2 TO DATA TO BE WRITTEN
17963                                   ;ENDDO
17964                                   ;GET FIRST ADDRESS OF 4K WORD DATA BUFFER
17965                                   ;CLEAR EXPECTED DATA
17966                                   ;DO UNTIL EXPECTED DATA EQUALS 20000(8)
17967                                   ;.      READ ADDRESS
17968                                   ;.      IF RECEIVED DATA NE EXPECTED DATA THEN
17969                                   ;.          GOTO DATA STORE PARITY ERROR ROUTINE
17970                                   ;.      ENDIF
17971                                   ;.      ADD 2 TO EXPECTED DATA
17972                                   ;.      ADD 2 TO ADDRESS
17973                                   ;ENDDO
17974                                   ;GET FIRST ADDRESS OF 8K BYTE DATA BUFFER
17975                                   ;CLEAR DATA TO BE WRITTEN
17976                                   ;DO UNTIL ALL BYTES CHECKED
17977                                   ;.      WRITE DATA TO ADDRESS
17978                                   ;.      ADD 1 TO ADDRESS
17979                                   ;.      ADD 1 TO DATA TO BE WRITTEN
17980                                   ;ENDDO
17981                                   ;GET FIRST ADDRESS OF 8K BYTE DATA BUFFER
17982                                   ;CLEAR EXPECTED DATA
17983                                   ;DO UNTIL EXPECTED DATA EQUALS 20000(8)
17984                                   ;.      READ ADDRESS
17985                                   ;.      IF RECEIVED DATA NE EXPECTED DATA THEN
17986                                   ;.          GOTO DATA STORE PARITY ERROR ROUTINE
17987                                   ;.      ENDIF
17988                                   ;.      ADD 1 TO EXPECTED DATA
17989                                   ;.      ADD 1 TO ADDRESS
17990                                   ;ENDDO
17991                                   ;ENDTST
17992                                   ;
17993                                   ;;*****************************************************************
     111600  000004                     TST23:  SCOPE
17994 111602  000240                             NOP
17995 111604  005737  003032                     TST     CCHPAS                  ;have done enough inclusive passes?
17996 111610  001002                             BNE     99$                     ; not yet
```

TEST - DATA STORE RAM TESTS

```
17997 111612  000240                        NOP                             ; debug aid
17998 111614  000550                        BR      TST24           ;;GO TO NEXT TEST
17999 111616  000240                99$:    NOP
18000
18001 111620  004737  136574                JSR     PC,     INITMM          ;SETUP MMU REGISTERS
18002 111624  012737  002000  172354        MOV     #2000,  KIPAR6          ;LET PAR6 MAP TO ADDRESS 200000
18003 111632  005237  177572                INC     SR0                     ;TURN ON MEMORY MANAGEMENT
18004 111636  012702  140000                MOV     #140000,R2              ;GET FIRST ADDRESS OF 4K WORD BUFFER
18005 111642  005001                        CLR     R1                      ;INIT DATA TO BE WRITTEN
18006 111644  052737  000002  177746        BIS     #BIT01, CCR             ;SET DIAG BIT IN CASE NO MEMORY PRESENT
18007 111652  042737  001000  177520        BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
18008 111660  020127  020000          1$:   CMP     R1,     #20000          ;DO UNTIL ALL ADDRESSES WRITTEN
18009 111664  001404                        BEQ     2$                      ;IF DONE GO CHECK DATA
18010 111666  010122                        MOV     R1,     (R2)+           ;WRITE DATA TO ADDRESS
18011 111670  062701  000002                ADD     #2,     R1              ;UPDATE DATA BY 2 (WORD BOUNDARY)
18012 111674  000771                        BR      1$                      ;ENDDO
18013 111676  042737  000002  177746  2$:   BIC     #BIT01, CCR             ;CLEAR DIAGNOSTIC BIT
18014 111704  012702  140000                MOV     #140000,R2              ;GET FIRST ADDRESS OF 4K WORD BUFFER
18015 111710  005001                        CLR     R1                      ;INIT DATA TO BE CHECKED
18016 111712  020127  020000          3$:   CMP     R1,     #20000          ;DO UNTIL ALL ADDRESSES CHECKED
18017 111716  001426                        BEQ     5$                      ;IF DONE GO DO BYTES
18018 111720  010137  001122                MOV     R1,     $BDADR          ;STORE FOR ERRORS
18019 111724  052737  100000  001122        BIS     #BIT15, $BDADR          ;
18020 111732  012237  114150                MOV     (R2)+,  RECDAT          ;READ TEST LOCATION
18021 111736  013737  177752  001160        MOV     HITMIS, $TMP0           ;STORE REGISTER
18022 111744  032737  000020  001160        BIT     #BIT04,$TMP0            ;HIT?
18023 111752  001001                        BNE     100$                    ;IF YES, BRANCH
18024 111754  104045                        ERROR   +45                     ;DATA RAM ERROR
18025 111756  020137  114150          100$: CMP     R1,     RECDAT          ;IF RECIEVED DATA NOT EQUAL TO EXPECTED
18026 111762  001401                        BEQ     4$                      ;THEN
18027 111764  104043                        ERROR   +43                     ;DATA RAM ERROR
18028 111766  062701  000002          4$:   ADD     #2,     R1              ;UPDATE EXPECTED DATA BY 2
18029 111772  000747                        BR      3$                      ;ENDDO
18030 111774  012702  140000          5$:   MOV     #140000,R2              ;GET FIRST ADDRESS OF 4K WORD BUFFER
18031 112000  005001                        CLR     R1                      ;INIT DATA TO BE WRITTEN
18032 112002  052737  000002  177746        BIS     #BIT01, CCR             ;SET DIAG BIT IN CASE NO MEMORY PRESENT
18033 112010  020127  020000          6$:   CMP     R1,     #20000          ;DO UNTIL ALL ADDRESSES WRITTEN
18034 112014  001404                        BEQ     7$                      ;IF DONE GO CHECK DATA
18035 112016  110122                        MOVB    R1,     (R2)+           ;WRITE DATA TO ADDRESS
18036 112020  062701  000001                ADD     #1,     R1              ;UPDATE DATA BY 1 (BYTE BOUNDARY)
18037 112024  000771                        BR      6$                      ;ENDDO
18038 112026  042737  000002  177746  7$:   BIC     #BIT01, CCR             ;CLEAR DIAGNOSTIC BIT
18039 112034  012702  140000                MOV     #140000,R2              ;GET FIRST ADDRESS OF 4K WORD BUFFER
18040 112040  005001                        CLR     R1                      ;INIT DATA TO BE CHECKED
18041 112042  020127  020000          8$:   CMP     R1,     #20000          ;DO UNTIL ALL ADDRESSES CHECKED
18042 112046  001426                        BEQ     10$                     ;IF DONE EXIT TEST
18043 112050  010137  001122                MOV     R1,     $BDADR          ;STORE FOR ERRORS
18044 112054  052737  100000  001122        BIS     #BIT15, $BDADR          ;
18045 112062  112237  114150                MOVB    (R2)+,  RECDAT          ;READ TEST LOCATION
18046 112066  013737  177752  001160        MOV     HITMIS, $TMP0           ;STORE REGISTER
18047 112074  032737  000020  001160        BIT     #BIT04,$TMP0            ;HIT?
18048 112102  001001                        BNE     101$                    ;IF YES, BRANCH
18049 112104  104045                        ERROR   +45                     ;DATA RAM ERROR
18050 112106  120137  114150          101$: CMPB    R1,     RECDAT          ;IF RECIEVED DATA NOT EQUAL TO EXPECTED
18051 112112  001401                        BEQ     9$                      ;THEN
18052 112114  104043                        ERROR   +43                     ;DATA RAM ERROR
18053 112116  062701  000001          9$:   ADD     #1,     R1              ;UPDATE EXPECTED DATA BY 1
```

TEST - DATA STORE RAM TESTS

```
 18054 112122  000747                        BR      8$                      ;ENDDO
 18055 112124  052737  001000  177520  10$:  BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
 18056 112132  005037  177572               CLR     SR0                     ;TURN OFF MMU
 18057
```

```
18059                           .SBTTL  TEST  - TAG STORE RAM TESTS
18060                           ;TAG STORE RAM TESTS - THERE ARE TWO TESTS FOR THE TAG STORE RAMS. THE FIRST
18061                           ;TEST IS AN ADDRESSING TEST TO ENSURE NO DUAL ADDRESSING OF THE TAG STORE.
18062                           ;THE SECOND TEST IS A "MOVING INVERSIONS" TEST THAT CHECKS THE DATA RELIABILITY
18063                           ;OF THE RAM STORE.
18064                           ;
18065                           ;DUAL ADDRESSING TEST - THIS WILL BE DONE BY FIRST FLUSHING THE CACHE, THEN
18066                           ;THE FIRST 512(10) LOCATIONS (BLOCK 0) WILL BE WRITTEN WITH ADDRESSES THAT
18067                           ;WILL CAUSE A INCREMENTING PATTERN TO BE STORED IN THOSE LOCATIONS OF THE TAG
18068                           ;STORE RAM. NEXT THE ENTIRE 4K ADDRESS RANGE WILL BE READ. THE HIT/MISS
18069                           ;REGISTER SHOULD ONLY REPORT HITS ON THE ADDRESSES THAT WERE ALLOCATED. THIS
18070                           ;PROCESS WILL BE REPEATED FOR EACH BLOCK.
18071                           ;
18072                           ;INITIALIZE LOW ADDRESS
18073                           ;SETUP AND ENABLE MMU
18074                           ;INITIALIZE BLOCK COUNTER
18075                           ;DO UNTIL ALL BLOCKS TESTED (8 TIMES)
18076                           ;.      FLUSH CACHE AND SET DIAGNOSTIC BIT
18077                           ;.      LET CURRENT ADDRESS = LOW ADDRESS
18078                           ;.      INITIALIZE ALLOCATION COUNTER
18079                           ;.      DO UNTIL ENTIRE BLOCK ALLOCATED (1000 TIMES)
18080                           ;.      .       .       READ CURRENT ADDRESS
18081                           ;.      .       ELSE
18082                           ;.      .       .       WRITE CURRENT ADDRESS
18083                           ;.      .       ENDIF
18084                           ;.      .       UPDATE CURRENT ADDRESS (ALSO ADD 200 TO PAR)
18085                           ;.      .       FOR I/O PAGE WRITE THE SAME ADDRESS AS BEFORE
18086                           ;.      ENDDO
18087                           ;.      INITIALIZE GOOD ADDRESS
18088                           ;.      INITIALIZE CURRENT ADDRESS
18089                           ;.      INITIALIZE LOCATION COUNTER
18090                           ;.      SAVE CONTENTS OF VECTOR 114
18091                           ;.      LET VECTOR 114 POINT TO TAG STORE PARITY ABORT ROUTINE
18092                           ;.      DO UNTIL ALL LOCATIONS CHECKED (1000 TIMES)
18093                           ;.      .       INITIALIZE CHECK COUNTER
18094                           ;.      .       DO UNTIL ADDRESS IN EACH BLOCK CHECKED
18095                           ;.      .       .       READ CURRENT ADDRESS
18096                           ;.      .       .       IF HIT THEN
18097                           ;.      .       .       .       IF CURRENT ADDRESS NE GOOD ADDRESS THEN
18098                           ;.      .       .       .       .       ERROR
18099                           ;.      .       .       .       ENDIF
18100                           ;.      .       .       ELSE
18101                           ;.      .       .       .       IF CURRENT ADDRESS EQUAL GOOD ADDRESS THEN
18102                           ;.      .       .       .       .       ERROR
18103                           ;.      .       .       .       ENDIF
18104                           ;.      .       .       ENDIF
18105                           ;.      .       .       UPDATE GOOD ADDRESS (ADD #2)
18106                           ;.      .       .       UPDATE CURRENT ADDRESS
18107                           ;.      .       ENDDO
18108                           ;.      .       UPDATE LOW ADDRESS (ADD #2000)
18109                           ;.      ENDDO
18110                           ;ENDDO
18111                           ;DISABLE MMU
18112                           ;RESTORE VECTOR 114
18113                           ;ENDTST
18114
18115                           ;;*****************************************************************
```

TEST - TAG STORE RAM TESTS

```
        112136  000004                  TST24:  SCOPE
  18116 112140  000240                          NOP
  18117 112142  005737  003032                  TST     CCHPAS                  ;have done enough inclusive passes?
  18118 112146  001003                          BNE     99$                     ; not yet
  18119 112150  000240                          NOP                             ; debug aid
  18120 112152  000137  112710                  JMP     18$                     ; yes skip this
  18121 112156  000240                  99$:    NOP
  18122
  18123 112160  013737  000004  001160          MOV     @#4,$TMPO               ;STORE TIMEOUT VECTOR
  18124 112166  012737  112712  000004          MOV     #20$,@#4                ;POINT NEW
  18125 112174  012737  140000  003016          MOV     #140000,LOWADD          ;INITIALIZE LOW ADDRESS (USE PAR6)
  18126 112202  004737  136574                  JSR     PC,     INITMM          ;INITIALIZE MMU
  18127 112206  005237  177572                  INC     SR0                     ;ENABLE MMU
  18128 112212  012737  000020  172516          MOV     #BIT04,MMR3             ;ENABLE 22-BIT MAPPING
  18129 112220  012737  177770  003154          MOV     #-10,   LOOPIN          ;DO UNTIL ALL BLOCKS TESTED
  18130 112226  012701  000000          1$:     MOV     #0,R1                   ;DO IN 2 WORDS TO AVOID PMI MEMORY
  18131 112232  042737  001000  177520  9$:     BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
  18132 112240  005037  172354                  CLR     KIPAR6                  ;SET UP PAR6 FOR THIS TEST
  18133 112244  012737  000402  177746          MOV     #402,   CCR             ;FLUSH CACHE AND SET DIAG BIT
  18134 112252  013737  003016  114170          MOV     LOWADD, CURADD          ;GET FIRST ADDRESS IN CURRENT BLOCK
  18135 112260  012737  177400  003152          MOV     #-400,  ALLCTR          ;DO UNTIL ALL ADDRESSES ALLOCATED
  18136 112266  022737  002000  172354  2$:     CMP     #2000,  KIPAR6          ;IF ADDRESS LESS THAN 32K
  18137 112274  002413                          BLT     3$                      ;THEN
  18138 112276  052737  100000  172314          BIS     #BIT15,KIPDR6           ;SET BYPASS
  18139 112304  017702  001660                  MOV     @CURADD,R2              ;STORE CURRENT DATA
  18140 112310  042737  100000  172314          BIC     #BIT15,KIPDR6           ;ALLOCATE NEXT ACCESS
  18141 112316  010277  001646                  MOV     R2,@CURADD              ;WRITE ALLOCATE
  18142 112322  000402                          BR      4$                      ;ELSE
  18143 112324  005077  001640          3$:     CLR     @CURADD                 ;WRITE CURRENT ADDRESS
  18144 112330  062737  000002  114170  4$:     ADD     #2,     CURADD          ;UPDATE CURRENT ADDRESS
  18145 112336  062737  000200  172354          ADD     #200,   KIPAR6
  18146 112344  022737  177600  172354          CMP     #177600,KIPAR6          ;REACHED I/O PAGE?
  18147 112352  001003                          BNE     10$                     ;BRANCH IF NOT
  18148 112354  162737  000200  172354          SUB     #200,KIPAR6             ;DON'T UPDATE PAR FOR I/O PAGE
  18149 112362  005237  003152          10$:    INC     ALLCTR                  ;IF ALL ADDRESSES ALLOCATED
  18150 112366  002737                          BLT     2$                      ;ENDDO
  18151 112370  052737  000004  177746          BIS     #BIT02, CCR             ;RUN WITH FORCE MISS
  18152 112376  013737  003016  003020          MOV     LOWADD, GOODAD          ;INITIALIZE GOOD ADDRESS
  18153 112404  060137  003020                  ADD     R1,     GOODAD          ;
  18154 112410  012737  140000  114170          MOV     #140000,CURADD          ;GET FIRST ADDRESS
  18155 112416  060137  114170                  ADD     R1,     CURADD          ;START WITH 1 OR 2 LOCATION
  18156 112422  005037  172354                  CLR     KIPAR6
  18157 112426  072127  000006                  ASH     #6,R1                   ;
  18158 112432  060137  172354                  ADD     R1,KIPAR6               ;MAKE SURE ON THE RIGHT BOUNDARY
  18159 112436  012737  177600  003152          MOV     #-200,  ALLCTR          ;DO UNTIL ALL LOCATIONS CHECKED
  18160 112444  012737  177770  114144  5$:     MOV     #-10,   DCOUNT          ;DO UNTIL ADDRESS IN EACH BLOCK CHECKED
  18161 112452  013737  114170  001122  6$:     MOV     CURADD, $BDADR          ;IN CASE OF ERRORS
  18162 112460  042737  160000  001122          BIC     #160000,$BDADR          ;CLEAR PAR BITS
  18163 112466  042737  000004  177746          BIC     #BIT02, CCR             ;CLEAR FORCE MISS
  18164 112474  005777  001470                  TST     @CURADD                 ;READ CURRENT ADDRESS
  18165 112500  013737  177752  114150          MOV     HITMIS, RECDAT          ;STORE REGISTER
  18166 112506  032737  000004  114150          BIT     #BIT2,  RECDAT          ;IF ACCESS WAS A HIT
  18167 112514  001406                          BEQ     7$                      ;THEN
  18168 112516  023737  114170  003020          CMP     CURADD, GOODAD          ;IF CURRENT ADDRESS NOT EQUAL TO GOOD
  18169 112524  001407                          BEQ     8$                      ;ADDRESS THEN
  18170 112526  104046                          ERROR   +46                     ;ERROR IN TAG STORE
  18171 112530  000405                          BR      8$                      ;
```

TEST - TAG STORE RAM TESTS

```
 18172 112532  023737  114170  003020  7$:     CMP     CURADD, GOODAD          ;IF CURRENT ADDRESS EQUAL GOOD ADDRESS
 18173 112540  001001                          BNE     8$                      ;THEN
 18174 112542  104047                          ERROR   +47                     ;ERROR IN TAG STORE
 18175 112544  062737  001000  114170  8$:     ADD     #1000,  CURADD          ;UPDATE TO NEXT BLOCK
 18176 112552  052737  000004  177746          BIS     #BIT02, CCR             ;RUN WITH FORCE MISS
 18177 112560  005237  114144                  INC     DCOUNT                  ;IF ADDRESS NOT CHECKED FOR EACH BLOCK
 18178 112564  002732                          BLT     6$                      ;ENDDO
 18179 112566  062737  000004  003020          ADD     #4,     GOODAD          ;UPDATE GOOD ADDRESS
 18180 112574  162737  007774  114170          SUB     #7774,  CURADD          ;INCREMENT IN 2 WORDS
 18181 112602  062737  000400  172354          ADD     #400,   KIPAR6
 18182 112610  005237  003152                  INC     ALLCTR                  ;IF ALL ADDRESSES CHECKED
 18183 112614  002713                          BLT     5$                      ;ENDDO
 18184 112616  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
 18185 112624  062701  000002                  ADD     #2,     R1              ;PREPARE FOR THE 2ND PASS THRU
 18186 112630  022701  000002                  CMP     #2,     R1              ;DONE TWICE?
 18187 112634  001002                          BNE     30$                     ;IF SO, EXIT
 18188 112636  000137  112232                  JMP     9$
 18189 112642  062737  002000  003016  30$:     ADD     #2000,  LOWADD          ;UPDATE TO NEXT BLOCK TO BE ALLOCATED
 18190 112650  005237  003154                  INC     LOOPIN                  ;IF ALL BLOCKS WERE TESTED
 18191 112654  002002                          BGE     15$                     ;
 18192 112656  000137  112226                  JMP     1$                      ;ENDDO
 18193 112662  012737  000400  177746  15$:     MOV     #400,   CCR             ;CLEAR DIAGNOSTIC BIT
 18194 112670  005037  177572                  CLR     SR0                     ;DISABLE MMU
 18195 112674  013737  001160  000004          MOV     $TMP0,@#4               ;RESTORE TIMEOUT VECTOR
 18196 112702  042737  000020  172516          BIC     #BIT04,MMR3             ;ENABLE 22-BIT MAPPING
 18197 112710                          18$:
       112710  000401                          BR      TST25           ;;GO TO NEXT TEST
 18198
 18199 112712  000002                  20$:    RTI
 18200
 18201
```

TEST - STANDALONE MODE TEST

```
18203                                   .SBTTL  TEST - STANDALONE MODE TEST
18204                                   ;THIS TEST VERIFIES THAT NMX CAN BE CREATED IN STANDALONE MODE.
18205                                   ;
18206                                   ;ALLOCATE INSTRUCTIONS IN CACHE
18207                                   ;GUARANTEE MISS ON TEST LOCATION
18208                                   ;IN STANALONE MODE ACCESS TEST LOCATION
18209                                   ;IF NO TIMEOUT THEN
18210                                   ;.      ERROR
18211                                   ;ENDIF
18212                                   ;
18213                                   ;;*****************************************************************
    112714  000004                      TST25:  SCOPE
18214 112716  000240                              NOP
18215 112720  005737  003032                      TST     CCHPAS                  ;have done enough inclusive passes?
18216 112724  001002                              BNE     99$                     ; not yet
18217 112726  000240                              NOP                             ; debug aid
18218 112730  000452                              BR      TST26           ;;GO TO NEXT TEST
18219 112732  000240                      99$:    NOP
18220
18221 112734  012737  000400  177746              MOV     #400,CCR                ;FLUSH CACHE
18222 112742  012701  112742                      MOV     #.,R1                   ;START WITH CURRENT INSTRUCTION
18223 112746  005721                      1$:     TST     (R1)+                   ;READ A WORD
18224 112750  022701  113046                      CMP     #10$,R1                 ;DONE?
18225 112754  001374                              BNE     1$                      ;IF NOT, CONTINUE
18226 112756  005737  020000                      TST     @#20000                 ;TO GUARANTY MISS ON 0
18227 112762  013702  000004                      MOV     @#4,R2                  ;STORE TIMEOUT VECTOR
18228 112766  012737  113022  000004              MOV     #5$,@#4                 ;POINT NEW TO THE TEST
18229 112774  012737  000340  000006              MOV     #340,@#6                ;AT PRIORITY 7
18230 113002  042737  001000  177520              BIC     #BIT09,BCSR             ;DISABLE HALT ON BREAK
18231 113010  052737  000400  177520              BIS     #BIT08,BCSR             ;GO TO STANDALONE MODE
18232 113016  005737  000000                      TST     @#0                     ;MISS, SHOULD TIMEOUT
18233 113022  042737  000400  177520  5$:         BIC     #BIT08,BCSR             ;CLEAR STANDALONE BIT
18234 113030  052737  001000  177520              BIS     #BIT09,BCSR             ;ENABLE HALT ON BREAK
18235 113036  022716  113022                      CMP     #5$,(SP)                ;TIMEOUT?
18236 113042  001401                              BEQ     10$                     ;IF YES, BRANCH
18237 113044  104044                              ERROR   +44
18238 113046  012706  001100          10$:        MOV     #1100,SP                ;RESTORE STACK
18239 113052  010237  000004                      MOV     R2,@#4                  ;AND TIMEOUT VECTOR
```

```
18241                           .SBTTL  TEST  - MOVING INVERSIONS TEST FOR DATA RAMS
18242                           ;MOVING INVERSIONS TEST FOR DATA RAMS - THE TEST IS STARTED AFTER LOADING THE
18243                           ;RAM STORE WITH 0'S. EACH ADDRESS IS READ AND VERIFIED TO BE ALL 0'S. THEN A
18244                           ;1 IS SUBSTITUTED IN A BIT POSITION AND THE NEW WORD IS WRITTEN. NEXT THE
18245                           ;ADDRESS IS READ TO VERIFY THE NEW CONTENTS. THIS IS REPEATED FOR EACH BIT OF
18246                           ;THE WORD LEAVING THE ARRAY FILED WITH 1'S. THE WHOLE PROCESS IS REPEATED
18247                           ;PLUGGING IN 0'S TO THE 1'S AND REPEATED TWICE MORE ADDRESSING IN THE DOWNWARD
18248                           ;DIRECTION. FINALLY EVERYTHING IS REPEATED FOR EACH BIT POSITION BEING THE
18249                           ;LSB. TO SAVE TIME AND KNOWING THE LAYOUT OF THE RAM CHIPS INSTEAD OF DOING
18250                           ;ONLY A SINGLE BIT AT A TIME EVERY FOURTH BIT WILL BE DONE CONCURRENTLY.
18251                           ;THIS TEST RUNS IN STANDALONE MODE.
18252                           ;
18253                           ;BGNTST
18254                           ;SETUP AND ENABLE MMU
18255                           ;SETUP CCR TO ABORT PARITY ERRORS
18256                           ;CLEAR CACHE
18257                           ;DO IN STANDALONE MODE FOR EACH HALF SEPARATELY
18258                           ;LET FWDSEQ = #1
18259                           ;LET ADDLSB = #1
18260                           ;DO UNTIL ADDLSB EQ #20000
18261                           ;.      LET CURDAT = 0
18262                           ;.      LET RITEDA = #1
18263                           ;.      LET NEWDAT = #1
18264                           ;.      IF FWDSEQ = #1 THEN
18265                           ;.      .       LET FSTADD EQUAL FIRST ADDRESS OF 4K BYTE BUFFER
18266                           ;.      .       LET LASTAD EQUAL LAST ADDRESS OF 4K BYTE BUFFER
18267                           ;.      ELSE
18268                           ;.      .       LET FSTADD EQUAL LAST ADDRESS OF 4K BYTE BUFFER
18269                           ;.      .       LET LASTAD EQUAL FIRST ADDRESS OF 4K BYTE BUFFER
18270                           ;.      ENDIF
18271                           ;.      LET CURADD = FSTADD
18272                           ;.      LET DCOUNT = #0
18273                           ;.      SAVE CONTENTS OF VECTOR 114
18274                           ;.      LET VECTOR 114 POINT TO DATA STORE PARITY ABORT ROUTINE
18275                           ;.      DO UNTIL DCOUNT EQ #10
18276                           ;.      .       LET RECDAT = @CURADD
18277                           ;.      .       IF RECDAT NE CURDAT THEN
18278                           ;.      .       .       LET R1 EQUAL CURRENT DATA
18279                           ;.      .       .       GOTO DATA STORE PARITY ERROR ROUTINE
18280                           ;.      .       ENDIF
18281                           ;.      .       IF DCOUNT GT #3 THEN
18282                           ;.      .       .       LET @CURADD = @CURADD CLEARBY RITEDA
18283                           ;.      .       ELSE
18284                           ;.      .       .       LET @CURADD = @CURADD SETBY RITEDA
18285                           ;.      .       ENDIF
18286                           ;.      .       LET RECDAT = @CURADD
18287                           ;.      .       IF RECDAT NE NEWDAT THEN
18288                           ;.      .       .       LET R1 EQUAL NEW DATA
18289                           ;.      .       .       GOTO DATA STORE PARITY ERROR ROUTINE
18290                           ;.      .       ENDIF
18291                           ;.      .       IF CURADD EQ LASTAD THEN
18292                           ;.      .       .       IF RITEDA = #210 THEN
18293                           ;.      .       .       .       IF DCOUNT NE #7 THEN
18294                           ;.      .       .       .       .       LET CURDAT = #377
18295                           ;.      .       .       .       .       LET RITEDA = #1
18296                           ;.      .       .       .       .       LET NEWDAT = #376
18297                           ;.      .       .       .       ENDIF
```

```
18298                                   ;.      .       .       ELSE
18299                                   ;.      .       .       .       LET CURDAT = NEWDAT
18300                                   ;.      .       .       .       ROTATE RITEDA
18301                                   ;.      .       .       .       IF DCOUNT GT #3 THEN
18302                                   ;.      .       .       .       .       LET NEWDAT = NEWDAT CLEAREDBY RITEDA
18303                                   ;.      .       .       .       ELSE
18304                                   ;.      .       .       .       .       LET NEWDAT = NEWDAT SETBY RITEDA
18305                                   ;.      .       .       .       ENDIF
18306                                   ;.      .       .       ENDIF
18307                                   ;.      .       .       INCREMENT DCOUNT
18308                                   ;.      .       .       LET CURADD = FSTADD
18309                                   ;.      .       ELSE
18310                                   ;.      .       .       IF FWDSEQ = #1 THEN
18311                                   ;.      .       .       .       LET CURADD = CURADD + ADDLSB
18312                                   ;.      .       .       .       IF CARRY THEN
18313                                   ;.      .       .       .       .       LET CURADD = CURADD + #1
18314                                   ;.      .       .       .       ENDIF
18315                                   ;.      .       .       ELSE
18316                                   ;.      .       .       .       LET CURADD = CURADD - ADDLSB
18317                                   ;.      .       .       .       IF CARRY THEN
18318                                   ;.      .       .       .       .       LET CURADD = LASTAD - ADDLSB
18319                                   ;.      .       .       .       .       LET CURADD = CURADD - #1
18320                                   ;.      .       .       .       ENDIF
18321                                   ;.      .       .       ENDIF
18322                                   ;.      .       ENDIF
18323                                   ;.      ENDDO
18324                                   ;.      IF FWDSEQ EQ #1 THEN
18325                                   ;.      .       LET FWDSEQ = #0
18326                                   ;.      ELSE
18327                                   ;.      .       ROTATE ADDLSB
18328                                   ;.      .       LET FWDSEQ = #1
18329                                   ;.      ENDIF
18330                                   ;ENDDO
18331                                   ;RESTORE VECTOR 114
18332                                   ;ENDTST
18333
18334                                   ;;*****************************************************************
     113056  000004                    TST26:  SCOPE
18335 113060  000240                            NOP
18336 113062  005737  003032                    TST     CCHPAS                  ;have done enough inclusive passes?
18337 113066  001003                            BNE     99$                     ; not yet
18338 113070  000240                            NOP                             ; debug aid
18339 113072  000137  114172                    JMP     ENDMOV                  ; yes skip this
18340 113076  000240                    99$:    NOP
18341
18342 113100  032777  000200  066032            BIT     #BIT07,@SWR             ;RUN THIS TEST?
18343 113106  001002                            BNE     100$                    ;IF SET, GO DO IT
18344 113110  000137  114172                    JMP     ENDMOV                  ;OTHERWISE, GO TO NEXT TEST
18345 113114  042737  001000  177520  100$:    BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
18346 113122  004737  136574                    JSR     PC,     INITMM          ;SETUP MEMORY MANAGEMENT
18347 113126  012737  002000  172354            MOV     #2000,KIPAR6            ;START ON 32K BOUNDARY
18348 113134  005237  177572                    INC     SR0                     ;TURN ON MMU
18349 113140  052737  000002  177746            BIS     #2,     CCR             ;SET DIAG. BIT
18350 113146  013737  000004  001160            MOV     @#4,    $TMP0           ;SAVE 4
18351                                   ;
18352                                   ;STORE TEST IN THE FIRST 2K AND THEN IN THE SECOND 2K
18353                                   ;
```

TEST - MOVING INVERSIONS TEST FOR DATA RAMS

```
18354 113154  012703  140000              MOV     #140000,R3          ;START FOR THE TEST
18355 113160  012704  150000              MOV     #150000,R4          ;LOWER BOUNDARY TEST AREA
18356 113164  012705  157777              MOV     #157777,R5          ;HIGH BOUNDARY TEST AREA
18357 113170  000406                      BR      2$
18358 113172  012703  150000        1$:   MOV     #150000,R3          ;START OF THE TEST
18359 113176  012704  140000              MOV     #140000,R4          ;LOWER BOUNDARY TEST AREA
18360 113202  012705  147777              MOV     #147777,R5          ;HIGH BOUNDARY
18361 113206  012702  113326        2$:   MOV     #STMOVI,R2          ;START WITH CURRENT
18362 113212  010300                      MOV     R3,R0               ;MOVE TO UPPER 4K
18363 113214  012220                3$:   MOV     (R2)+,(R0)+         ;WORD BY WORD
18364 113216  022702  114172              CMP     #ENDMOV,R2          ;ALL DONE
18365 113222  001374                      BNE     3$                  ;
18366 113224  010400                      MOV     R4,R0               ;CLEAR CACHE UNDER TEST
18367 113226  012701  004000              MOV     #4000,  R1
18368 113232  005020                4$:   CLR     (R0)+
18369 113234  077102                      SOB     R1,     4$
18370 113236  004713                      JSR     PC,(R3)             ;GO DO THE ROUTINE
18371 113240  005702                      TST     R2                  ;ANY ERRORS?
18372 113242  001411                      BEQ     5$                  ;IF NOT, CONTINUE
18373 113244  010037  114150              MOV     R0,RECDAT           ;DATA RECEIVED
18374 113250  010237  001122              MOV     R2,$BDADR           ;ADDRESS RECIEVED
18375 113254  042737  140000  001122      BIC     #140000,$BDADR      ;STRIP OF PAR BITS
18376 113262  104043                      ERROR   +43                 ;
18377 113264  000403                      BR      6$                  ;EXIT
18378 113266  022703  150000        5$:   CMP     #150000,R3          ;DONE FOR BOTH HALVES?
18379 113272  001337                      BNE     1$                  ;IF NOT, DO AGAIN
18380 113274  052737  001000  177520  6$:   BIS     #BIT09,BCSR         ;ENABLE HALT ON BREAK
18381 113302  013737  001160  000004      MOV     $TMP0,@#4           ;RESTORE TIMEOUT VECTOR
18382 113310  012737  000400  177746      MOV     #400,CCR            ;INIT CCR FOR EXIT
18383 113316  005037  177572              CLR     SR0                 ;TURN OFF MMU
18384 113322  000137  114172              JMP     ENDMOV              ;GO TO NEXT TEST
18385
18386                             .DSABL AMA
18387 113326  052737  000400  177520  STMOVI: BIS     #BIT08,@#BCSR       ;STANDALONE MODE
18388 113334  005002                      CLR     R2                  ;ERROR INDICATOR
18389 113336  012767  000001  000606      MOV     #1,     FWDSEQ      ;INIT UPWARD ADDRESSING INDICATOR
18390 113344  012767  000001  000602      MOV     #1,     ADDLSB      ;INIT LSB AND DO LOOP UNTIL SHIFTED OUT
18391 113352  042737  100000  172300      BIC     #100000,@#KIPDR0    ;NO BYPASS
18392 113360  012737  172360  000004      MOV     #KDPAR0,@#4         ;ALLOCATE TIMEOUT VECTOR
18393 113366  012737  000340  000006      MOV     #340,   @#6         ;AT PRIORITY 7
18394 113374  012737  000006  172360      MOV     #6,     @#KDPAR0    ;PUT RETURN
18395 113402  052737  100000  172300      BIS     #100000,@#KIPDR0    ;BYPASS
18396 113410  005067  000546        TSTLUP: CLR     CURDAT              ;INIT CURRENT DATA
18397 113414  012767  000021  000534      MOV     #21,    RITEDA      ;INIT DATA TO BE WRITTEN
18398 113422  012767  000021  000530      MOV     #21,    NEWDAT      ;INIT EXPECTED DATA
18399 113430  005767  000516              TST     FWDSEQ              ;IF ADDRESSING UPWARD
18400 113434  001405                      BEQ     1$                  ;THEN
18401 113436  010467  000522              MOV     R4,FSTADD           ;LET FIRST ADDRESS EQUAL LOWEST VALUE
18402 113442  010567  000520              MOV     R5,LSTADD           ;LET LAST ADDRESS EQUAL HIGHEST VALUE
18403 113446  000404                      BR      2$                  ;ELSE
18404 113450  010567  000510        1$:   MOV     R5,FSTADD           ;LET FIRST ADDRESS EQUAL HIGHEST VALUE
18405 113454  010467  000506              MOV     R4,LSTADD           ;LET LAST ADDRESS EQUAL LOWEST VALUE
18406 113460  016767  000500  000502  2$:   MOV     FSTADD, CURADD      ;LET CURRENT ADDRESS EQUAL FIRST ADDRESS
18407 113466  005067  000452              CLR     DCOUNT              ;INIT LOOP COUNTER
18408 113472  022767  000010  000444  BGNTLP: CMP     #10,    DCOUNT      ;DO LOOP 8 TIMES (4 TIMES TO WRITE 1'S
18409 113500  001567                      BEQ     ENDTLP              ;AND 4 TIMES TO WRITE BACK 0'S)
18410                                     ;
```

TEST - MOVING INVERSIONS TEST FOR DATA RAMS

```
18411                                   ; DON'T REWRITE TIMEOUT VECTOR
18412                                   ;
18413 113502  016700  000462                    MOV     CURADD, R0          ;STORE CURRENT ADDRESS
18414 113506  042700  170000                    BIC     #170000,R0          ;LEAVE ONLY LOW 4K BITS
18415 113512  022700  000004                    CMP     #4,     R0          ;TIMOUT VECTOR?
18416 113516  001531                          BEQ     8$                  ;IF SO, DON'T REWRITE IT
18417 113520  022700  000005                    CMP     #5,     R0          ;
18418 113524  001526                          BEQ     8$                  ;
18419 113526  022700  000006                    CMP     #6,     R0          ;
18420 113532  001523                          BEQ     8$                  ;
18421 113534  022700  000007                    CMP     #7,     R0          ;
18422 113540  001520                          BEQ     8$                  ;
18423                                   ;
18424                                   ;CHECK PATTERNS NOW
18425                                   ;
18426 113542  016701  000414                    MOV     CURDAT, R1          ;MOVE GOOD DATA TO R1
18427 113546  117767  000416  000374            MOVB    @CURADD,RECDAT      ;FIRST READ LOCATION
18428 113554  126767  000370  000400            CMPB    RECDAT, CURDAT      ;IF RECIEVED DATA NOT EQUAL TO EXPECTED
18429 113562  001402                          BEQ     1$                  ;DATA THEN
18430 113564  000167  000334                    JMP     EXBAD               ;EXIT TEST
18431 113570  016701  000364          1$:       MOV     NEWDAT, R1          ;MOVE GOOD DATA TO R1
18432 113574  022767  000003  000342            CMP     #3,     DCOUNT      ;IF LOOP COUNT GREATER THAN 3
18433 113602  002004                          BGE     2$                  ;THEN
18434 113604  146777  000346  000356            BICB    RITEDA, @CURADD     ;CLEAR TEST DATA BY WRITE DATA
18435 113612  000403                          BR      3$                  ;ELSE
18436 113614  156777  000336  000346  2$:       BISB    RITEDA, @CURADD     ;SET TEST DATA BY WRITE DATA
18437 113622  117767  000342  000320  3$:       MOVB    @CURADD,RECDAT      ;DO READ AFTER WRITE
18438 113630  126767  000314  000322            CMPB    RECDAT, NEWDAT      ;IF RECIEVED DATA NOT EQUAL TO EXPECTED
18439 113636  001402                          BEQ     4$                  ;DATA THEN
18440 113640  000167  000260                    JMP     EXBAD               ;EXIT TEST
18441 113644  026767  000320  000314  4$:       CMP     CURADD, LSTADD      ;IF CURRENT ADDRESS EQUALS LAST ADDRESS
18442 113652  001053                          BNE     8$                  ;THEN
18443 113654  022767  000210  000274            CMP     #210,   RITEDA      ;AND IF WRITE PATTERN EQUALS LAST
18444 113662  001016                          BNE     5$                  ;PATTERN THEN
18445 113664  022767  000007  000252            CMP     #7,     DCOUNT      ;AND TEST IS NOT ON LAST LOOP
18446 113672  001435                          BEQ     7$                  ;THEN
18447 113674  012767  000377  000260            MOV     #377,   CURDAT      ;SET UP TO WRITE 0'S
18448 113702  012767  000021  000246            MOV     #21,    RITEDA
18449 113710  012767  000356  000242            MOV     #356,   NEWDAT
18450 113716  000423                          BR      7$                  ;ELSE
18451 113720  016767  000234  000234  5$:       MOV     NEWDAT, CURDAT      ;SET UP FOR NEXT DATA PATTERN
18452 113726  000241                          CLC
18453 113730  106167  000222                    ROLB    RITEDA
18454 113734  005567  000216                    ADC     RITEDA              ;GET A PATTERN
18455 113740  022767  000003  000176            CMP     #3,     DCOUNT      ;AND IF DOWNWARD ADDRESSING
18456 113746  002004                          BGE     6$                  ;THEN
18457 113750  046767  000202  000202            BIC     RITEDA, NEWDAT      ;LET NEWDAT BE CLEARED BY WRITE DATA
18458 113756  000403                          BR      7$                  ;ELSE
18459 113760  056767  000172  000172  6$:       BIS     RITEDA, NEWDAT      ;LET NEWDAT BE SET BY WRITE DATA
18460 113766  005267  000152          7$:       INC     DCOUNT              ;UPDATE LOOP COUNTER
18461 113772  016767  000166  000170            MOV     FSTADD, CURADD      ;REINIT FIRST ADDRESS FOR THIS PASS
18462 114000  000426                          BR      10$                 ;ELSE
18463 114002  005767  000144          8$:       TST     FWDSEQ              ;IF ADDRESSING UPWARD
18464 114006  001412                          BEQ     9$                  ;THEN
18465 114010  066767  000140  000152            ADD     ADDLSB, CURADD      ;CALCULATE NEXT HIGHER ADDRESS
18466 114016  020567  000146                    CMP     R5,CURADD           ;IF CURRENT ADDRESS HAS BEEN INCREASED
18467 114022  002015                          BGE     10$                 ;ABOVE HIGHEST ADDRESS THEN
```

TEST - MOVING INVERSIONS TEST FOR DATA RAMS

```
18468 114024  162767  007777  000136          SUB     #7777,  CURADD          ;ROLL ADDRESS BACK
18469 114032  000411                          BR      10$                     ;ELSE
18470 114034  166767  000114  000126  9$:     SUB     ADDLSB, CURADD          ;CALCULATE NEXT LOWER ADDRESS
18471 114042  020467  000122                  CMP     R4,CURADD               ;IF CURRENT ADDRESS HAS BEEN DECREASED
18472 114046  003403                          BLE     10$                     ;BELOW LOWEST ADDRESS THEN
18473 114050  062767  007777  000112          ADD     #7777,  CURADD          ;ROLL ADDRESS BACK
18474 114056  000605                  10$:    BR      BGNTLP                  ;ENDDO
18475 114060  005767  000066          ENDTLP: TST     FWDSEQ                  ;IF ADDRESSING UPWARD FINISHED
18476 114064  001404                          BEQ     1$                      ;THEN
18477 114066  005067  000060                  CLR     FWDSEQ                  ;DO ADDRESSING DOWNWARD
18478 114072  000167  177312                  JMP     TSTLUP
18479 114076  012767  000001  000046  1$:     MOV     #1,     FWDSEQ          ;SET ADDRESSING UPWARD INDICATOR
18480 114104  006167  000044                  ROL     ADDLSB                  ;UPDATE LSB TO NEXT POSITION
18481 114110  022767  020000  000036  ENDLUP: CMP     #20000,ADDLSB           ;ALL DONE?
18482 114116  001406                          BEQ     EXITST                  ;ENDDO
18483 114120  000167  177264                  JMP     TSTLUP
18484 114124  016700  000020          EXBAD:  MOV     RECDAT,R0               ;STORE RECEIVED DATA
18485 114130  016702  000034                  MOV     CURADD,R2               ;STORE ADDRESS
18486 114134  042737  000400  177520  EXITST: BIC     #BIT08,@#BCSR           ;OUT OF STANDALONE
18487 114142  000207                          RTS     PC
18488                                 .ENABL AMA
18489
18490 114144  000000                  DCOUNT: .WORD   0
18491 114146  000000                  EXPDAT: .WORD   0               ;STORES EXPECTED (GOOD) DATA FOR COMPARISONS
18492 114150  000000                  RECDAT: .WORD   0               ;STORES RECIEVED DATA TO BE VERIFIED
18493 114152  000000                  FWDSEQ: .WORD   0               ;USED TO INDICATE DIRECTION OF ADDRESSING
18494 114154  000000                  ADDLSB: .WORD   0               ;STORES LEAST SIGNIFICANT BIT FOR RAM TESTS
18495 114156  000000                  RITEDA: .WORD   0               ;STORES WRITE DATA FOR RAM TESTS
18496 114160  000000                  NEWDAT: .WORD   0               ;DATA STORE FOR RAM TESTS
18497 114162  000000                  CURDAT: .WORD   0               ;DATA STORE FOR RAM TESTS
18498 114164  000000                  FSTADD: .WORD   0               ;STORES FIRST ADDRESS IN ADDRESSING SEQUENCE
18499 114166  000000                  LSTADD: .WORD   0               ;STORES LAST ADDRESS IN ADDRESSING SEQUENCE
18500 114170  000000                  CURADD: .WORD   0               ;STORES CURRENT ADDRESS FOR RAM TESTS
18501
18502 114172                          ENDMOV:
```

```
18504                                   .SBTTL  TEST  - MOVING INVERSIONS TEST FOR TAG STORE
18505                                   ;MOVING INVERSIONS TEST - THE TEST IS STARTED AFTER LOADING THE RAM STORE
18506                                   ;WITH 0'S. EACH ADDRESS IS READ AND VERIFIED TO BE ALL 0'S. THEN A 1 IS
18507                                   ;SUBSTITUTED IN A BIT POSITION AND THE NEW WORD IS WRITTEN. NEXT THE ADDRESS
18508                                   ;IS READ TO VERIFY THE NEW CONTENTS. THIS IS REPEATED FOR EACH BIT OF THE
18509                                   ;WORD LEAVING THE ARRAY FILED WITH 1'S. THE WHOLE PROCESS IS REPEATED PLUGGING
18510                                   ;IN 0'S TO THE 1'S AND REPEATED TWICE MORE ADDRESS IN THE DOWARD DIRECTION.
18511                                   ;FINALLY EVERYTHING IS REPEATED FOR EACH BIT POSITION BEING THE LSB. TO SAVE
18512                                   ;TIME AND KNOWING THE LAYOUT OF THE RAM CHIPS INSTEAD OF DOING ONLY A SINGLE
18513                                   ;BIT AT A TIME EVERY THIRD BIT WILL BE DONE CONCURRENTLY. ALSO NOTE THAT
18514                                   ;SINCE THE TAG STORE CANNOT BE DIRECTLY ACCESSED THE ENTIRE PATTERN MUST BE
18515                                   ;DONE BY DOING MEMORY CYCLES TO THE CORRECT BUS ADDRESSES. TO DO THIS THE
18516                                   ;TEST WILL BE DONE WITH THE DIAGNOSTIC BIT IN THE CCR (BIT 1) SET TO A 1.
18517                                   ;THIS TEST RUNS IN STANDALONE MODE.
18518                                   ;
18519                                   ;
18520                                   ;BGNTST
18521                                   ;SETUP AND ENABLE MMU
18522                                   ;SETUP CCR TO ABORT PARITY ERRORS
18523                                   ;DO IN STANDALONE MODE FOR EACH HALF SEPARATELY
18524                                   ;LET FWDSEQ = #1
18525                                   ;LET ADDLSB = #1
18526                                   ;DO UNTIL ADDLSB EQ #20000
18527                                   ;.      LET NEWDAT = 22200
18528                                   ;.      LET CURDAT = 0
18529                                   ;.      IF FWDSEQ = #1 THEN
18530                                   ;.      .       LET FSTADD EQUAL FIRST ADDRESS OF 4K BYTE BUFFER
18531                                   ;.      .       LET LASTAD EQUAL LAST ADDRESS OF 4K BYTE BUFFER
18532                                   ;.      ELSE
18533                                   ;.      .       LET FSTADD EQUAL LAST ADDRESS OF 4K BYTE BUFFER
18534                                   ;.      .       LET LASTAD EQUAL FIRST ADDRESS OF 4K BYTE BUFFER
18535                                   ;.      ENDIF
18536                                   ;.      LET DCOUNT = #0
18537                                   ;.      DO UNTIL DCOUNT EQ #10
18538                                   ;.      .       READ CURADD USING CURDAT AS PAR
18539                                   ;.      .       IF MISS THEN
18540                                   ;.      .       .       ERROR
18541                                   ;.      .       ENDIF
18542                                   ;.      .       WRITE CURADD USING NEWDAT AS PAR
18543                                   ;.      .       ENDIF
18544                                   ;.      .       IF HIT THEN
18545                                   ;.      .       .       ERROR
18546                                   ;.      .       ENDIF
18547                                   ;.      .       READ CURADD USING NEWDAT AS PAR
18548                                   ;.      .       IF MISS THEN
18549                                   ;.      .       .       ERROR
18550                                   ;.      .       ENDIF
18551                                   ;.      .       IF CURADD EQ LASTAD THEN
18552                                   ;.      .       .       LET CURDAT = NEWDAT
18553                                   ;.      .       .       IF DCOUNT < #1 THEN
18554                                   ;.      .       .       .       LET NEWDAT = NEWDAT SETBY RITEDA ROTATED LEFT
18555                                   ;.      .       .       ENDIF
18556                                   ;.      .       .       IF DCOUNT > #1 THEN
18557                                   ;.      .       .       .       LET NEWDAT = NEWDAT CLR BY RITEDA ROTATED LEFT
18558                                   ;.      .       .       ELSE
18559                                   ;.      .       .       .       LET NEWDAT = #155400 (NO ALL 1'S)
18560                                   ;.      .       .       .       LET RITEDA = #22200
```

TEST - MOVING INVERSIONS TEST FOR TAG STORE

```
18561                                   ;.      .       .       ENDIF
18562                                   ;.      .       .       INCREMENT DCOUNT
18563                                   ;.      .       ELSE
18564                                   ;.      .       .       IF FWDSEQ = #1 THEN
18565                                   ;.      .       .       .       LET CURADD = CURADD + ADDLSB
18566                                   ;.      .       .       .       IF CURADD GE HIGH ADDRESS THEN
18567                                   ;.      .       .       .       .       LET CURADD = CURADD + #7776
18568                                   ;.      .       .       .       ENDIF
18569                                   ;.      .       .       ELSE
18570                                   ;.      .       .       .       LET CURADD = CURADD - ADDLSB
18571                                   ;.      .       .       .       IF CURADD LE LOW ADDRESS THEN
18572                                   ;.      .       .       .       .       LET CURADD = CURADD - #7776
18573                                   ;.      .       .       .       ENDIF
18574                                   ;.      .       .       ENDIF
18575                                   ;.      .       ENDIF
18576                                   ;.      ENDDO
18577                                   ;.      IF FWDSEQ EQ #1 THEN
18578                                   ;.      .       LET FWDSEQ = #0
18579                                   ;.      ELSE
18580                                   ;.      .       ROTATE ADDLSB
18581                                   ;.      .       LET FWDSEQ = #1
18582                                   ;.      ENDIF
18583                                   ;ENDDO
18584                                   ;RESTORE PARITY ABORT VECTOR
18585                                   ;ENDTST
18586
18587                                   ;;*****************************************************************************
        114172  000004                  TST27:  SCOPE
18588   114174  000240                          NOP
18589   114176  005737  003032                  TST     CCHPAS                  ;have done enough inclusive passes?
18590   114202  001003                          BNE     99$                     ; not yet
18591   114204  000240                          NOP                             ; debug aid
18592   114206  000137  115314                  JMP     ENDTAG                  ; yes skip this
18593   114212  000240                  99$:    NOP
18594
18595   114214  032777  000100  064716          BIT     #BIT06,@SWR             ;RUN THIS TEST?
18596   114222  001002                          BNE     100$                    ;IF SET, GO DO IT
18597   114224  000137  115314                  JMP     ENDTAG                  ;OTHERWISE, GO TO NEXT TEST
18598   114230  042737  001000  177520  100$:   BIC     #1000,@CSR              ;DISABLE HALT ON BREAK
18599   114236  004737  136574                  JSR     PC,     INITMM          ;SETUP MEMORY MANAGEMENT
18600   114242  005237  177572                  INC     SR0                     ;TURN ON MMU
18601   114246  012737  000020  172516          MOV     #BIT04,MMR3             ;ENABLE 22-BIT MAPPING
18602   114254  052737  000002  177746          BIS     #2,     CCR             ;SET DIAG. BIT
18603   114262  013737  000004  001160          MOV     @#4,    $TMP0           ;STORE TIMEOUT VECTOR
18604                                   ;
18605                                   ;STORE TEST IN THE FIRST 2K AND THEN IN THE SECOND 2K
18606                                   ;
18607   114270  012703  140000                  MOV     #140000,R3              ;START FOR THE TEST
18608   114274  012704  130000                  MOV     #130000,R4              ;LOWER BOUNDARY TEST AREA
18609   114300  012705  137776                  MOV     #137776,R5              ;HIGH BOUNDARY TEST AREA
18610   114304  000406                          BR      2$
18611   114306  012703  150000          1$:     MOV     #150000,R3              ;START OF THE TEST
18612   114312  012704  120000                  MOV     #120000,R4              ;LOWER BOUNDARY TEST AREA
18613   114316  012705  127776                  MOV     #127776,R5              ;HIGH BOUNDARY
18614   114322  012737  002000  172354  2$:     MOV     #2000,KIPAR6            ;ABOVE 32K
18615   114330  012702  114134                  MOV     #EXITST,R2              ;START WITH CURRENT
18616   114334  010300                          MOV     R3,R0                   ;MOVE TO UPPER 4K
```

TEST - MOVING INVERSIONS TEST FOR TAG STORE

```
18617 114336  012220                    3$:     MOV     (R2)+,(R0)+             ;WORD BY WORD
18618 114340  022702  115314                    CMP     #ENDTAG,R2              ;ALL DONE?
18619 114344  001374                            BNE     3$                      ;
18620 114346  012700  114450                    MOV     #STMOVT,R0              ;ADDRESS OF ROUTINE
18621 114352  162700  114134                    SUB     #EXITST,R0              ;PROPER OFSET
18622 114356  050003                            BIS     R0,R3                   ;
18623 114360  004713                            JSR     PC,(R3)                 ;GO DO THE ROUTINE
18624 114362  005700                            TST     R0                      ;ANY ERRORS?
18625 114364  001407                            BEQ     4$                      ;CONTINUE
18626 114366  010037  001122                    MOV     R0,$BDADR               ;STORE FAILED ADDRESS
18627 114372  042737  160000  001122            BIC     #160000,$BDADR          ;CLEAR PAR
18628 114400  104050                            ERROR   +50
18629 114402  000403                            BR      5$                      ;EXIT TEST
18630 114404  022703  150000              4$:   CMP     #150000,R3              ;DONE FOR BOTH HALVES?
18631 114410  103736                            BLO     1$                      ;IF NOT, DO AGAIN
18632 114412  013737  001160  000004    5$:     MOV     $TMP0,  @#4             ;RESTORE TIMEOUT VECTOR
18633 114420  012737  000400  177746            MOV     #400,   CCR             ;INIT CCR FOR EXIT
18634 114426  005037  177572                    CLR     SR0                     ;TURN OFF MMU
18635 114432  005037  172516                    CLR     MMR3                    ;
18636 114436  052737  001000  177520            BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
18637 114444  000137  115314                    JMP     ENDTAG                  ;EXIT TEST
18638
18639                                   .DSABL AMA
18640 114450  052737  000400  177520  STMOVT: BIS     #BIT08,@#BCSR           ;STANDALONE MODE
18641 114456  042737  100000  172312            BIC     #BIT15, @#KIPDR5        ;NO BYPASS
18642 114464  042737  100000  172300            BIC     #100000,@#KIPDR0        ;NO BYPASS
18643 114472  012737  172360  000004            MOV     #KDPAR0,@#4             ;ALLOCATE TIMEOUT VECTOR
18644 114500  012737  000340  000006            MOV     #340,   @#6             ;AT 7
18645 114506  012737  000006  172360            MOV     #6,     @#KDPAR0        ;PUT RETURN
18646 114514  052737  100000  172300            BIS     #100000,@#KIPDR0        ;BYPASS
18647 114522  005037  172352                    CLR     @#KIPAR5                ;
18648 114526  010400                            MOV     R4,R0                   ;CLEAR CACHE UNDER TEST
18649 114530  012701  004000                    MOV     #4000,  R1              ;
18650 114534  005020                    4$:     CLR     (R0)+                   ;
18651 114536  077102                            SOB     R1,     4$              ;
18652 114540  005000                            CLR     R0                      ;CLEAR ERROR FLAG
18653 114542  012767  000001  177402            MOV     #1,     FWDSEQ          ;SET UPWARD ADDRESSING INDICATOR
18654 114550  012767  000002  177376            MOV     #2,     ADDLSB          ;INITIALIZE LEAST SIGNIFIGANT BIT
18655 114556  005067  177400            TSLOOP: CLR     CURDAT                  ;OLD DATA
18656 114562  012767  022200  177370            MOV     #22200,NEWDAT           ;SET ADDRESS BITS
18657 114570  012767  022200  177360            MOV     #22200,RITEDA           ;SET ADDRESS BITS
18658 114576  005767  177350                    TST     FWDSEQ                  ;IF ADDRESSING UPWARD
18659 114602  001405                            BEQ     1$                      ;THEN
18660 114604  010467  177354                    MOV     R4,     FSTADD          ;FIRST ADDRESS WILL BE LOWEST ADDRESS
18661 114610  010567  177352                    MOV     R5,     LSTADD          ;AND LAST ADDRESS WILL BE HIGHEST
18662 114614  000404                            BR      2$                      ;ELSE
18663 114616  010567  177342              1$:   MOV     R5,     FSTADD          ;FIRST ADDRESS WILL BE HIGHEST ADDRESS
18664 114622  010467  177340                    MOV     R4,     LSTADD          ;AND LAST ADDRESS WILL BE LOWEST
18665 114626  005067  177312              2$:   CLR     DCOUNT                  ;INITIALIZE LOOP COUNTER
18666 114632  016767  177326  177330            MOV     FSTADD, CURADD
18667                                   ;
18668                                   ; DON'T REWRITE TIMEOUT VECTOR
18669                                   ;
18670 114640  016700  177324              3$:   MOV     CURADD, R0              ;STORE CURRENT ADDRESS
18671 114644  042700  170000                    BIC     #170000,R0              ;LEAVE ONLY LOW 4K BITS
18672 114650  022700  000004                    CMP     #4,     R0              ;TIMOUT VECTOR?
18673 114654  001526                            BEQ     16$                     ;IF SO, DON'T REWRITE IT
```

TEST - MOVING INVERSIONS TEST FOR TAG STORE

```
18674 114656  022700  000006                CMP   #6,     R0          ;OR PRIORITY
18675 114662  001523                        BEQ   16$                 ;
18676 114664  016737  177272  172352        MOV   CURDAT, @#KIPAR5    ;GET OLD PATTERN
18677 114672  005777  177272                TST   @CURADD             ;OLD DATA OK?
18678 114676  013767  177752  177244        MOV   @#HITMIS,RECDAT
18679 114704  032767  000004  177236  5$:   BIT   #BIT02, RECDAT      ;IF ACCESS WAS A MISS
18680 114712  001002                        BNE   6$                  ;THEN
18681 114714  000167  000346                JMP   EXBAD2              ;ERROR, EXIT
18682 114720  016737  177234  172352  6$:   MOV   NEWDAT, @#KIPAR5    ;GET NEW PATTERN
18683 114726  005077  177236                CLR   @CURADD             ;REGISTER AND WRITE LOCATION
18684 114732  013767  177752  177210        MOV   @#HITMIS,RECDAT
18685 114740  032767  000004  177202  8$:   BIT   #BIT02, RECDAT      ;IF ACCESS WAS A HIT
18686 114746  001406                        BEQ   9$                  ;THEN
18687 114750  032767  000010  177172  10$:  BIT   #BIT03, RECDAT      ;MISS?
18688 114756  001402                        BEQ   9$                  ;IF SO, CONTINUE
18689 114760  000167  000302                JMP   EXBAD2              ;ERROR
18690 114764  005777  177200          9$:   TST   @CURADD             ;REGISTER AND READ LOCATION
18691 114770  013767  177752  177152        MOV   @#HITMIS,RECDAT
18692 114776  032767  000004  177144  11$:  BIT   #BIT02, RECDAT      ;IF ACCESS WAS A MISS
18693 115004  001002                        BNE   12$                 ;THEN
18694 115006  000167  000254                JMP   EXBAD2              ;ERROR, EXIT
18695 115012  026767  177150  177150  12$:  CMP   LSTADD, CURADD      ;IF CURRENT ADDRESS IS LAST ADDRESS
18696 115020  001044                        BNE   16$                 ;THEN
18697 115022  016767  177132  177132        MOV   NEWDAT,CURDAT       ;NEW KIPAR5
18698 115030  022767  000001  177106        CMP   #1,     DCOUNT      ;IF LOOP COUNTER LESS THAN 1
18699 115036  003406                        BLE   13$                 ;THEN
18700 115040  006367  177112                ASL   RITEDA              ;UPDATE OF NEW DATA....
18701 115044  056767  177106  177106        BIS   RITEDA, NEWDAT      ;BY SETTING BITS
18702 115052  000421                        BR    15$                 ;ENDIF
18703 115054  022767  000001  177062  13$:  CMP   #1,     DCOUNT      ;CHANGE PATTERN?
18704 115062  001007                        BNE   14$                 ;IF SO, BRANCH
18705 115064  012767  022200  177064        MOV   #22200, RITEDA      ;START WITH THE SAME
18706 115072  012767  155400  177060        MOV   #155400,NEWDAT      ;DON'T DO ALL 1'S
18707 115100  000406                        BR    15$                 ;
18708 115102  006367  177050          14$:  ASL   RITEDA              ;UPDATE OF NEW DATA....
18709 115106  046767  177044  177044        BIC   RITEDA, NEWDAT      ;BY CLEARING BITS
18710 115114  000400                        BR    15$                 ;ELSE
18711 115116  005267  177022          15$:  INC   DCOUNT              ;INCREMENT THE LOOP COUNTER
18712 115122  016767  177036  177040        MOV   FSTADD, CURADD      ;FINISH FIRST CURRENT ADDRESS
18713 115130  000426                        BR    18$                 ;ELSE
18714 115132  005767  177014          16$:  TST   FWDSEQ              ;IF ADDRESSING UPWARD
18715 115136  001412                        BEQ   17$                 ;THEN
18716 115140  066767  177010  177022        ADD   ADDLSB, CURADD      ;ADD THE VALUE OF THE CURRENT LSB
18717 115146  020567  177016                CMP   R5,     CURADD      ;IF CURRENT ADDRESS GREATER THAN HIGHEST
18718 115152  002015                        BGE   18$                 ;VIRTUAL ADDRESS THEN
18719 115154  162767  007776  177006        SUB   #7776,  CURADD      ;ROLL IT BACK
18720 115162  000411                        BR    18$                 ;ELSE
18721 115164  166767  176764  176776  17$:  SUB   ADDLSB, CURADD      ;IF ADDRESSING DOWNWARD THEN SUBTRACT LSB
18722 115172  020467  176772                CMP   R4,     CURADD      ;IF CURRENT ADDRESS LESS THEN LOWEST
18723 115176  003403                        BLE   18$                 ;VIRTUAL ADDRESS THEN
18724 115200  062767  007776  176762        ADD   #7776,  CURADD      ;ROLL IT BACK
18725 115206  022767  000005  176730  18$:  CMP   #5,     DCOUNT      ;IF LOOP COUNTER LESS THAN 7
18726 115214  003402                        BLE   181$                ;THEN LOOP BACK FOR NEXT BIT POSITION
18727 115216  000167  177416                JMP   3$
18728 115222  005767  176724          181$: TST   FWDSEQ              ;IF ADDRESSING UPWARD
18729 115226  001404                        BEQ   19$                 ;THEN
18730 115230  005067  176716                CLR   FWDSEQ              ;START ADDRESSING DOWNWARD
```

TEST - MOVING INVERSIONS TEST FOR TAG STORE

```
18731 115234  000167  177316                    JMP     TSLOOP                  ;ELSE
18732 115240  012767  000001  176704  19$:      MOV     #1,     FWDSEQ          ;START ADDRESSING UPWARD
18733 115246  006167  176702                    ROL     ADDLSB                  ;ROTATE LSB TO NEXT POSITION
18734 115252  022767  020000  176674            CMP     #20000, ADDLSB          ;ALL DONE?
18735 115260  001406                            BEQ     TSEND                   ;EXIT
18736 115262  000167  177270                    JMP     TSLOOP                  ;ENDDO
18737 115266  013701  172352          EXBAD2:   MOV     @#KIPAR5,R1             ;STORE PAR
18738 115272  016700  176672                    MOV     CURADD, R0              ;AND ADDRESS
18739 115276  012737  001200  172352  TSEND:    MOV     #1200,  @#KIPAR5        ;RESTORE PAR
18740 115304  042737  000400  177520            BIC     #BIT08, @#BCSR          ;OUT OF STANDALONE MODE
18741 115312  000207                            RTS     PC                      ;RETURN
18742 115314                          ENDTAG:
18743                                 .ENABL AMA
18744
```

```
 18746                                  .SBTTL  TEST  - PCR READ/WRITE BITS
 18747                                  ;PCR AND BCSR READ/WRITE BITS
 18748                                  ;THE FIRST TEST WILL CHECK THAT PCR REGISTER IS BOTH WORD AND
 18749                                  ;BYTE ADDRESSABLE. BITS 14-09 AND 06-01 WILL BE WRITTEN AND READ
 18750                                  ;AS ZEROES AND ONES. THE REST OF THE BITS HAVE TO BE ALL 0'S.
 18751                                  ;ROUTINE TEST
 18752                                  ;.      SAVE PCR
 18753                                  ;.      LET PCR=0
 18754                                  ;.      DO FOR PATTERN=001111,110011,101010,010101
 18755                                  ;.      .       WRITE PCR<14-09>=PATTERN
 18756                                  ;.      .       WRITE PCR<06-01>=PATTERN
 18757                                  ;.      .       IF PCR<14-09> NE PATTERN OR PCR<06-01> NE PREVIOUS
 18758                                  ;.      .                                              PATTERN
 18759                                  ;.      .       .       THEN ERROR
 18760                                  ;.      .       ENDIF
 18761                                  ;.      .       WRITE PCR<06-01>=PATTERN
 18762                                  ;.      .       IF PCR<14-09> NE PATTERN OR PCR<06-01> NE PATTERN
 18763                                  ;.      .       .       THEN ERROR
 18764                                  ;.      .       ENDIF
 18765                                  ;.      ENDDO
 18766                                  ;.      WRITE PCR=0101010101010101
 18767                                  ;.      IF PCR NE 0101010001010100 THEN
 18768                                  ;.      .       ERROR
 18769                                  ;.      ENDIF
 18770                                  ;ENDROUTINE
 18771
 18772                                  ;;*****************************************************************
     115314  000004                     TST30:  SCOPE
 18773 115316  005037  177522                   CLR     PCR                     ;INITIALIZE PCR TO 0
 18774 115322  012702  115461                   MOV     #SIXBIT+1,R2            ;R2->TABLE OF PATTERNS FOR 6 R/W BITS IN EAC
H BYTE
 18775 115326  012704  115460                   MOV     #SIXBIT,R4              ;R3 POINTER TO PREVIOUS PATTERN
 18776 115332  012703  000004                   MOV     #4,R3                   ;DO 4 TIMES
 18777                                  ;
 18778                                  ; WRITE TO HIGH BYTE FIRST
 18779                                  ;
 18780 115336  111237  177523           1$:     MOVB    (R2),PCR+1              ;WRITE TO HIGH BYTE
 18781 115342  121237  177523                   CMPB    (R2),PCR+1              ;BYTE WRITTEN OK?
 18782 115346  001003                           BNE     2$                      ;IF NOT, BRANCH
 18783 115350  121437  177522                   CMPB    (R4),PCR                ;LOW BYTE CHANGED?
 18784 115354  001405                           BEQ     3$                      ;IF NOT, BRANCH
 18785 115356  111237  001125           2$:     MOVB    (R2),$GDDAT+1           ;EXPECTED PATTERN HIGH BYTE
 18786 115362  111437  001124                   MOVB    (R4),$GDDAT             ;LOW BYTE
 18787 115366  104051                           ERROR   +51                     ;ERROR PCR READ/WRITE BITS
 18788 115370  105724                   3$:     TSTB    (R4)+                   ;INCREMENT POINTER FOR OLD
 18789                                  ;
 18790                                  ; WRITE TO LOW BYTE
 18791                                  ;
 18792 115372  111237  177522                   MOVB    (R2),PCR                ;WRITE TO LOW BYTE
 18793 115376  121237  177522                   CMPB    (R2),PCR                ;BYTE WRITTEN OK?
 18794 115402  001003                           BNE     4$                      ;IF NOT,BRANCH
 18795 115404  121237  177522                   CMPB    (R2),PCR                ;HIGH BYTE CHANGED?
 18796 115410  001405                           BEQ     5$                      ;IF NOT, BRANCH
 18797 115412  111237  001124           4$:     MOVB    (R2),$GDDAT             ;EXPECTED PATTERN HIGH BYTE
 18798 115416  111237  001124                   MOVB    (R2),$GDDAT             ;LOW BYTE
 18799 115422  104051                           ERROR   +51                     ;ERROR PCR READ/WRITE BITS
 18800 115424  105722                   5$:     TSTB    (R2)+                   ;INCREMENT POINTER FOR NEW PATTERN
 18801 115426  077335                           SOB     R3,1$                   ;DO FOR ALL 4 PATTERNS
```

```
18802                                   ;
18803                                   ; NOW TRY WORD ADDRESSING
18804                                   ;
18805 115430  012737  052525  177522          MOV     #52525,PCR              ;WRITE A PATERN
18806 115436  022737  052124  177522          CMP     #52124,PCR              ;ALL BUT BITS <8,7,0> OK?
18807 115444  001404                          BEQ     6$                      ;IF SO, BRANCH
18808 115446  112737  052124  001124          MOVB    #52124,$GDDAT           ;EXPECTED PATTERN
18809 115454  104051                          ERROR   +51                     ;ERROR PCR READ/WRITE BITS
18810 115456                            6$:
      115456  000403                          BR      TST31           ;;GO TO NEXT TEST
18811
18812 115460     000     036     146  SIXBIT: .BYTE 0,36,146,124,52           ;001111,110011,101010,010101
18813                                                                           ;BINARY DIVIDE FOR 6 BITS
18814                                   .EVEN
18815
```

```
                                .SBTTL  TEST  - BCSR READ/WRITE BITS
                                ;THE SECOND TEST WILL CHECK THAT BCSR<7-5;2-0> CAN BE WRITTEN AND
                                ;READ AS ZEROES AND ONES.  BCSR<14,03> SHOULD BE 0'S.  BCSR<04>
                                ;SHOULD BE CLEARED BY RESET INSTRUCTION.
                                ;ROUTINE TEST
                                ;.      FOR PATTERN=011,010,101 DO
                                ;.      .       WRITE BCSR<7-5>=PATTERN
                                ;.      .       IF BCSR<7-5> NE PATTERN THEN
                                ;.      .       .       ERROR
                                ;.      .       ENDIF
                                ;.      .       WRITE BCSR<2-0>=PATTERN
                                ;.      .       IF BCSR<2-0> NE PATTERN THEN
                                ;.      .       .       ERROR
                                ;.      .       ENDIF
                                ;.      ENDDO
                                ;.      IF BCSR<14,03> NE <0,0> THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;.      LET BCSR<04>=1
                                ;.      IF BCSR<04> NE #1 THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;.      EXECUTE "RESET"
                                ;.      IF BCSR<04> NE 0 THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;.      LET BCSR<04>=0 (THIS BIT IS WRITE ENABLE FOR EAROM)
                                ;ENDROUTINE

                                ;;*****************************************************************
 115466  000004                 TST31:  SCOPE
18847 115470  013737  177520  002730        MOV     BCSR,SAVBR              ;SAVE BCSR
18848 115476  005037  177520              CLR     BCSR                    ;CLEAR BCSR
                                ;
                                ; WRITE TO BITS <7-5> AND <2-0>
                                ;
18852 115502  012703  115726              MOV     #THRBIT,R3              ;POINTER FOR PATTERN TABLE
18853 115506  005037  001124              CLR     $GDDAT                  ;CLEAR A LOCATION
18854 115512  012702  000003              MOV     #3,R2                   ;DO FOR ALL PATTERNS
18855 115516  111337  177520      1$:     MOVB    (R3),BCSR               ;WRITE TO BITS <7-5>
18856 115522  111337  001124              MOVB    (R3),$GDDAT             ;EXPECTED PATTERN
18857 115526  122337  177520              CMPB    (R3)+,BCSR              ;BITS WRITTEN OK?
18858 115532  001401                      BEQ     2$                      ;IF SO, BRANCH
18859 115534  104052                      ERROR   +52                     ;ERROR IN BCSR READ/WRITE BITS
18860 115536  111337  177520      2$:     MOVB    (R3),BCSR               ;WRITE TO BITS <2-0>
18861 115542  111337  001124              MOVB    (R3),$GDDAT             ;EXPECTED PATTERN FOR ERRORS
18862 115546  122337  177520              CMPB    (R3)+,BCSR              ;BITS WRITTEN OK?
18863 115552  001401                      BEQ     3$                      ;IF SO, BRANCH
18864 115554  104052                      ERROR   +52                     ;ERROR IN BCSR READ/WRITE BITS
18865 115556  077221              3$:     SOB     R2,1$                   ;CONTINUE TILL ALL PATTERNS DONE
                                ;
                                ; CHECK UNUSED BITS <3>
                                ;
18869 115560  012737  000010  177520        MOV     #10,BCSR                ;WRITE TO BIT <3>
18870 115566  032737  000010  177520        BIT     #BIT03,BCSR             ;ALL ZEROES?
18871 115574  001403                      BEQ     4$                      ;IF YES, BRANCH
18872 115576  005037  001124              CLR     $GDDAT                  ;EXPECTED PATTERN
```

TEST - BCSR READ/WRITE BITS

```
18873 115602  104052                          ERROR   +52                             ;ERROR IN BCSR READ/WRITE BITS
18874                                 ;
18875                                 ; CHECK THAT BIT <4> CLEARS BY RESET
18876                                 ;
18877 115604  052737  000020  177520  4$:     BIS     #BIT04,BCSR                     ;SET BIT 4
18878 115612  032737  000020  177520          BIT     #BIT04,BCSR                     ;WRITTEN OK?
18879 115620  001005                        BNE     5$                              ;IF SO BRANCH
18880 115622  012737  000020  001124        MOV     #BIT04,$GDDAT                   ;EXPECTED PATTERN
18881 115630  104052                        ERROR   +52                             ;ERROR IN BCSR READ/WRITE BITS
18882 115632  000415                        BR      7$                              ;EXIT TEST
18883 115634  042737  000020  177520  5$:     BIC     #BIT04,BCSR                     ;TRY TO CLEAR BIT 4
18884 115642  032737  000020  177520          BIT     #BIT04,BCSR                     ;CLEARED OK?
18885 115650  001403                        BEQ     6$                              ;IF SO BRANCH
18886 115652  005037  001124                CLR     $GDDAT                          ;EXPECTED PATTERN
18887 115656  104052                        ERROR   +52                             ;ERROR IN BCSR READ/WRITE BITS
18888 115660  005737  001206          6$:     TST     $PASS                           ;FIRST PASS?
18889 115664  001014                        BNE     8$                              ;IF  NOT FIRST PASS,EXIT
18890 115666  052737  000020  177520  7$:     BIS     #BIT04,BCSR                     ;SET BIT 4 AGAIN
18891 115674  000005                        RESET                                   ;EXECUTE RESET
18892 115676  032737  000020  177520          BIT     #BIT04,BCSR                     ;BIT 4 CLEARED?
18893 115704  001404                        BEQ     8$                              ;IF YES, BRANCH
18894 115706  104053                        ERROR   +53                             ;RESET DOESN'T CLEAR BCSR<4>
18895 115710  042737  000020  177520          BIC     #BIT04,BCSR                     ;CLEAR BIT 4
18896 115716  013737  002730  177520  8$:     MOV     SAVBR,BCSR                      ;RESTORE BCSR
18897 115724  000403                        BR      TST32           ;;GO TO NEXT TEST
18898
18899 115726     140     003     100  THRBIT: .BYTE   140,3,100,2,240,5               ;011,010,101 FOR BITS <7-5,2-0>
18900
```

```
                                .SBTTL  TEST  - 16 BIT ROM CHECKSUM TEST
                                ;ROM'S CHECKSUMS
                                ;
                                ;16 BIT ROM TEST
                                ;THE FIRST TEST WILL CLEAR BCSR<07>, LOAD PCR<14-09> WITH
                                ;ROM ADDRESS BITS <14-09>, AND CHECK CHECKSUMS OF 16-BIT ROM
                                ;BY ACCESSING IT THRU BUS ADDRESSES 173000-173776.  THEN WITH
                                ;BCSR<06;05> BOTH CLEAR, PCR<06-01> USED AS ADDRESS BITS 14-09,
                                ;THE SAME THING WILL BE DONE BY ADDRESSING 16-BIT ROM THRU BUS ADDRESSES
                                ;165000-165776. THE RESULTS SHOULD BE THE SAME AND SHOULD COMPARE
                                ;WITH THAT STORED IN THE BOOT AND DIAGNOSTIC ROM.
                                ;
                                ;BCSR    <07>    DISABLE 17773000
                                ;        <06>    DISABLE 17765000
                                ;        <05>    ROM SOCKET 3 AT 17765000
                                ;
                                ;ROUTINE TEST
                                ;.      LET BCSR<7,6,5>=0,0,0
                                ;.      DO FOR R1 FROM #0 TO #31. BY #1 DO
                                ;.      .       LET PCR<14-09>=R1
                                ;.      .       DO FOR R2 FROM #0 TO #776 BY 2
                                ;.      .       .       CALCULATE CHECKSUM THRU 173000
                                ;.      .       ENDDO
                                ;.      ENDDO
                                ;.      IF CHECKSUM NE #0 THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;.      DO FOR R1 FROM #0 TO #31. BY #1
                                ;.      .       LET PCR<06-01>=R1
                                ;.      .       DO FOR R2 FROM #0 TO #776 BY 2
                                ;.      .       .       CALCULATE CHECKSUM THRU 165000
                                ;.      .       ENDDO
                                ;.      ENDDO
                                ;.      IF CHECKSUM NE #0 THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;ENDROUTINE
                                ;
                                ;;*****************************************************************
                                TST32:  SCOPE
                                        MOV     BCSR,SAVBR                      ;SAVE BCSR
                                        BIC     #BIT07!BIT06!BIT05,BCSR         ;READ 16 BIT ROM
                                        BIS     #1000,BCSR                      ; enable HOB, for APT
                                ;
                                ; CALCULATE LOW BYTE CHECKSUM'S THRU 173000 AND 165000
                                ;
                                        CLR     R1                              ;PAGE COUNT FOR ALL 8K
                                        CLR     PCR                             ;CLEAR PAGE CONTROL REGISTER
                                1$:     CLR     ACTCHS                          ;CLEAR CHECKSUM AT 173000
                                        CLR     $TMPO                           ;AT 165000
                                        CLR     R2                              ;CLEAR COUNTER THRU A PAGE
                                2$:     MOV     173000(R2),R3                   ;GET LOW BYTE THRU 173000
                                        MOV     165000(R2),R4                   ;THRU 165000
                                        ADD     R3,ACTCHS                       ;CALCULATE CHECKSUM THRU 173000
                                        ADD     R4,$TMPO                        ;CALCULATE CHEKCSUM THRU 165000
                                        TST     (R2)+                           ;GET NEXT WORD
                                        CMP     #776,R2                         ;WORD BEFORE LAST?
```

```
18958 116026  001004                      BNE     3$              ;IF NOT, BRANCH
18959 116030  020337  177522              CMP     R3,PCR          ;PAGE NUMBER STORED OK?
18960 116034  001401                      BEQ     3$              ;IF YES, BRANCH
18961 116036  104054                      ERROR   +54             ;PAGE NUMBER STORED WRONG
18962 116040  022702  001000        3$:   CMP     #1000,R2        ;LAST WORD IN A PAGE?
18963 116044  003355                      BGT     2$              ;IF NOT, BRANCH
18964 116046  005737  002724              TST     ACTCHS          ;CHECKSUM 0?
18965 116052  001401                      BEQ     4$              ;IF YES, BRANCH
18966 116054  104054                      ERROR   +54             ;IN CHECKSUM AT 173000
18967 116056  005737  001160        4$:   TST     $TMP0           ;CHECKSUM 0 AT 165000?
18968 116062  001401                      BEQ     5$              ;IF YES, BRANCH
18969 116064  104054                      ERROR   +54             ;IN CHECKSUM AT 165000
18970 116066  062701  000002        5$:   ADD     #2,R1           ;GET NEW PAGE
18971 116072  110137  177522              MOVB    R1,PCR          ;STORE IN PCR<6-1>
18972 116076  110137  177523              MOVB    R1,PCR+1        ;STORE IN PCR<14-9>
18973 116102  022701  000160              CMP     #160,R1         ;ALL  PAGES IN R2 FROM BIT1?
18974 116106  003327                      BGT     1$              ;IF NOT BRANCH
18975                               ;
18976                               ; CHECK THE LAST 2K OF ASCII TEXT
18977                               ;
18978 116110  005037  002724        LASTCH: CLR     ACTCHS          ;CLEAR CHECKSUM AT 173000
18979 116114  005037  001160              CLR     $TMP0           ;AT 165000
18980 116120  005002                1$:   CLR     R2              ;CLEAR COUNTER THRU A PAGE
18981 116122  016203  173000        2$:   MOV     173000(R2),R3   ;GET LOW BYTE THRU 173000
18982 116126  016204  165000              MOV     165000(R2),R4   ;THRU 165000
18983 116132  060337  002724              ADD     R3,ACTCHS       ;CALCULATE CHECKSUM THRU 173000
18984 116136  060437  001160              ADD     R4,$TMP0        ;CALCULATE CHEKCSUM THRU 165000
18985 116142  005722                      TST     (R2)+           ;GET NEXT WORD
18986 116144  022702  001000        3$:   CMP     #1000,R2        ;LAST WORD IN A PAGE?
18987 116150  003364                      BGT     2$              ;IF NOT, BRANCH
18988 116152  022701  000176              CMP     #176,R1         ;LAST PAGE?
18989 116156  001005                      BNE     4$              ;IF NOT, BRANCH
18990 116160  023737  173774  177522      CMP.    173774,PCR      ;PROPER PAGE NUMBER?
18991 116166  001401                      BEQ     4$              ;IF YES, BRANCH
18992 116170  104054                      ERROR   +54             ;ERROR IN ROM
18993 116172  062701  000002        4$:   ADD     #2,R1           ;GET NEW PAGE
18994 116176  110137  177522              MOVB    R1,PCR          ;STORE IN PCR<6-1>
18995 116202  110137  177523              MOVB    R1,PCR+1        ;STORE IN PCR<14-9>
18996 116206  022701  000200              CMP     #200,R1         ;ALL  PAGES IN R2 FROM BIT1?
18997 116212  003342                      BGT     1$              ;IF NOT BRANCH
18998 116214  005737  002724              TST     ACTCHS          ;CHECKSUM 0?
18999 116220  001401                      BEQ     5$              ;IF YES, BRANCH
19000 116222  104054                      ERROR   +54             ;IN CHECKSUM AT 173000
19001 116224  005737  001160        5$:   TST     $TMP0           ;CHECKSUM 0 AT 165000?
19002 116230  001401                      BEQ     6$              ;IF YES, BRANCH
19003 116232  104054                      ERROR   +54             ;IN CHECKSUM AT 165000
19004 116234  005037  177522        6$:   CLR     PCR             ;CLEAR PCR
19005 116240  042737  001000  177520      BIC     #1000, BCSR     ; reset HOB for APT
19006 116246  013737  177520  002730      MOV     BCSR,SAVBR      ;RESTORE BCSR
19007
19008
19009
```

```
19011                                   .SBTTL  TEST  - 8 BIT EEROM CHECKSUM (105dec bytes) TEST
19012                                   ;THIS TEST WILL CLEAR BCSR<6>, SET BCSR<5>, LOAD PCR<5-1>
19013                                   ;WITH ADDRESS BITS 13-09, AND CHECK CRC PATTERNS OF 8-BIT EEROM BY
19014                                   ;ACCESSING IT THRU ADDRESSES 165000-165776. THIS TEST VERIFIES THE
19015                                   ;RESPONSE ONLY OF THE BASE AREA.
19016                                   ;ROUTINE TEST
19017                                   ;.      LET BCSR<5>=1
19018                                   ;.      LET OLDCRC=#0
19019                                   ;.      LET PCR<05-01>=#0
19020                                   ;.      CALCULATE CHECKSUM FOR THE FIRST 320 LOCATIONS
19021                                   ;.      IF RESULTING CHECKSUM NOT ZERO THEN
19022                                   ;.      .       ERROR
19023                                   ;.      ENDIF
19024                                   ;ENDROUITNE
19025
19026                                   ;;*****************************************************************
     116254  000004                     TST33:  SCOPE
19027 116256  013737  177520  002730            MOV     BCSR,SAVBR              ;SAVE BCSR
19028 116264  042737  000100  177520            BIC     #BIT06,BCSR             ;ENABLE INTERNAL RESPONSE
19029 116272  052737  000040  177520            BIS     #BIT05,BCSR             ;SELECT 8-BIT ROM
19030 116300  005037  001160                    CLR     $TMP0                   ;CLEAR SUM
19031                                   ;
19032                                   ; CALCULATE LOW BYTE CHECKSUM'S THRU 165000
19033                                   ;
19034 116304  005001                    1$:     CLR     R1                      ;PAGE COUNT FOR ALL 8K
19035 116306  005037  177522                    CLR     PCR                     ;CLEAR PAGE CONTROL REGISTER
19036 116312  005037  002724            2$:     CLR     ACTCHS                  ;CLEAR CHECKSUM AT 165000
19037 116316  005002                            CLR     R2                      ;CLEAR COUNTER THRU A PAGE
19038 116320  116204  165000            3$:     MOVB    165000(R2),R4           ;GET A BYTE THRU 165000
19039 116324  060437  001160                    ADD     R4,$TMP0                ;CALCULATE CHEKCSUM THRU 165000
19040 116330  005722                            TST     (R2)+                   ;GET NEXT WORD
19041 116332  022702  000316                    CMP     #316,R2                 ;WORD BEFORE LAST?
19042 116336  001004                            BNE     4$                      ;IF NOT, BRANCH
19043 116340  122704  000252                    CMPB    #252,R4                 ;314 SHOULD HAVE 252
19044 116344  001765                            BEQ     3$                      ;IF YES, BRANCH
19045 116346  104055                            ERROR   +55                     ;PAGE NUMBER STORED WRONG
19046 116350  022702  000322            4$:     CMP     #322,R2                 ;LAST WORD IN A PAGE?
19047 116354  003361                            BGT     3$                      ;IF NOT, BRANCH
19048 116356  113737  165010  001162            MOVB    165010,$TMP1            ;STORE SIZE,<3>=1 2K
19049 116364  105737  001160                    TSTB    $TMP0                   ;CHECKSUM 0 AT 165000?
19050 116370  001401                            BEQ     5$                      ;IF YES, BRANCH
19051 116372  104055                            ERROR   +55                     ;IN CHECKSUM AT 165000
19052 116374  005037  177522            5$:     CLR     PCR
19053 116400  013737  002730  177520            MOV     SAVBR,BCSR              ;RESTORE BCSR
19054
```

```
                        .SBTTL  TEST - 8-BIT EAROM READ-WRITE - TEST

                        ; This test writes a checkerboard pattern of alternating 1's and 0's in
                        ; 8-bit EAROM.
                        ; All base area 316 octal locations are saved. (105 dec plus 4 customer loc.)
                        ; It will run only if bit 12 is set in software switch register.
                        ; This is the eerom test enable, setable at runtime by an operator.  Loop
                        ; on error is prohibited, as it would be possible to destroy the EEROM
                        ; in a relativly short time with repeated write cycles.  The option to
                        ; continue beyond a single error is selected by by asserting the halt on
                        ; error switch at which time the operator may truncate the test.
                        ;
                        ; The switches which could affect the EEROM's life are masked out and
                        ; restored in the test immediatly following this one. <<<<<*********
                        ;
                        ; This test will be not be done if there is an APT system present, or if
                        ; bit #12 of the Software Switch register is not set.  If Bit 12 is set and
                        ; there is no APT system present (or in load mode), the EEROM test will be
                        ; run 1 time, all subsequent passes will not include the EEROM test.
                        ;
                        ;
                        ;                               -- SECTION 1
                        ;+  SAVE BASE AREA
                        ;+
                        ;+
                        ;+
                        ;+ TEST                         -- SECTION 2
                        ;+
                        ;+ 2K PROCEDURE
                        ;+    do while NOT$DONE$2000
                        ;+       Write 10101010 (252 oct)
                        ;+       delay
                        ;+       read  10101010 (252 oct)
                        ;+       update error list if incorrect read
                        ;+    enddo
                        ;+    do while NOT$DONE$2000
                        ;+       write 01010101 (125 oct)
                        ;+       delay
                        ;+       read  01010101 (125 oct)
                        ;+       update error list
                        ;+    enddo
                        ;+
                        ; CLEANUP -- SECTION 3
                        ; communicate failure mode and loc info
                        ; restore base area (105) + 4 customer locations
                        ; restore registers
                        ; RESTORE BACK BASE AREA
                        ;
                        ;
                        ;*******************************************************************************

                        ;;*****************************************************************
116406  000004          TST34:  SCOPE
19109 116410  017737  062524  003050        MOV     @SWR,SAVSWR             ;SAVE SOFTSWITCH SETTINGS
19110 116416  005737  001206                TST     $PASS                   ;do only once, iff all cond. O.K.
19111 116422  001006                        BNE     1$                      ;  on first pass ( 0)
```

```
19112 116424  000240                          NOP                             ; no, do againa
19113                                 ;       CMPB    #APTENV,$ENV            ;IN APT MODE?
19114                                 ;       BEQ     1$                      ; <IF YES, EXIT TEST>
19115 116426  032777  010000  062504          BIT     #BIT12,@SWR             ; bit 12 (do EEROM test) set?
19116 116434  001016                          BNE     2$                      ;
19117 116436  000240                          NOP
19118 116440  000137  116770          1$:     JMP     TSTEND                  ; exit point for program
19119 116444     015     012     116  33$:    .ASCIZ  <15><12>/NOW TESTING EEROM/<15><12>
19120                                         .EVEN
19121                                 ; save base area, switch registers
19122
19123 116472  104401  116444          2$:     TYPE    ,33$                    ; INDICATE EEROM TEST UNDERWAY
19124 116476  017737  062436  003050          MOV     @SWR,SAVSWR             ; save softswitch settings
19125 116504  042777  041777  062426          BIC     #41777, @SWR            ; clear any interferring
19126 116512  013737  177520  002730          MOV     BCSR,SAVBR              ;SAVE REGISTER
19127 116520  052737  001060  177520          BIS     #1060,BCSR              ;ENABLE INTERNAL ROM'S
19128 116526  000240                          NOP                             ;ENABLE 8-BIT ROM, HOB
19129 116530  012701  143762                  MOV     #$POWER+10,R1           ;LAST LOCATION IN PROGRAM
19130 116534  005037  177522                  CLR     PCR                     ;START WITH PAGE 0
19131 116540  005002                          CLR     R2                      ;DISPLACEMENT 0
19132 116542  016221  165000          3$:     MOV     165000(R2),(R1)+        ;STORE A WORD
19133 116546  005722                          TST     (R2)+                   ;GET NEXT WORD
19134 116550  022702  000334                  CMP     #334,R2                 ;ALL 332 LOCATIONS DONE?
19135 116554  003372                          BGT     3$                      ;IF NOT, CONTINUE
19136
19137                                 ; test 2K section
19138
19139 116556  112703  000252                  MOVB    #252, R3                ; first pattern, 10 101 010
19140 116562  005037  003022                  CLR     ERRCNT                  ; zero the cumulative error count
19141
19142 116566  000240                  201$:   NOP
19143 116570  005037  177522                  CLR     PCR                     ; clears page register (start @ 165000)
19144 116574  000240                  202$:   NOP
19145 116576  005002                          CLR     R2                      ; first location each page of EEROM
19146 116600  000240                  203$:   NOP
19147 116602  110362  165000                  MOVB    R3, 165000(R2)          ; write the test pattern
19148 116606  004737  116754                  JSR     PC, DELAY               ; wait for write time
19149 116612  120362  165000                  CMPB    R3,165000(R2)           ; read back the written word
19150 116616  001413                          BEQ     204$                    ; if O. K. readback, skip handle error
19151
19152 116620  005237  003022                  INC     ERRCNT                  ; update cumulative error count
19153 116624  116237  165000  001126          MOVB    165000(R2),$BDDAT       ; put the read data in display area
19154 116632  013704  177522                  MOV     PCR,R4
19155 116636  010237  001122                  MOV     R2,$BDADR               ; also address location info
19156 116642  104134                          ERROR   +134                    ; do EEROM read/write error report
19157 116644  000240                          NOP                             ; continue testing
19158
19159 116646  005722                  204$:   TST     (R2)+
19160 116650  022702  000776                  CMP     #776,R2                 ; last location checked in a page
19161 116654  003351                          BGT     203$                    ; continue till all page loc. tested
19162 116656  062737  000002  177522          ADD     #2,PCR                  ; change PCR every 512 dec. bytes
19163 116664  122737  000020  177522          CMPB    #20,PCR                 ; 8 (256 Byte pgs in 2K) * 2 (holes)=20
19164 116672  001340                          BNE     202$                    ; finish page
19165 116674  122703  000125                  CMPB    #125,R3                 ; test for both patterns written
19166 116700  001403                          BEQ     205$                    ; Exit point
19167 116702  112703  000125                  MOVB    #125,R3                 ; invert the test pattern
19168 116706  000727                          BR      201$                    ; do the test over with new pattern
```

TEST - 8-BIT EAROM READ-WRITE - TEST

```
19169
19170
19171 116710  000240                  205$:   NOP                             ; test over
19172
19173 116712  012701  143762                  MOV     #$POWER+10,R1           ;LAST LOCATION IN PROGRAM
19174 116716  005037  177522                  CLR     PCR                     ;START WITH PAGE 0
19175 116722  005002                          CLR     R2                      ;DISPLACEMENT 0
19176 116724  012162  165000          11$:    MOV     (R1)+,165000(R2)        ;RESTORE A WORD
19177 116730  004737  116754                  JSR     PC, DELAY               ; wait for write time
19178 116734  005722                          TST     (R2)+                   ;GET NEXT WORD
19179 116736  022702  000334                  CMP     #334,R2                 ;ALL 332 LOCATIONS DONE?
19180 116742  003370                          BGT     11$                     ;IF NOT, CONTINUE
19181 116744  013737  002730  177520          MOV     SAVBR,BCSR              ;RESTORE REGISTER
19182 116752  000406                          BR      TSTEND                  ; goto next test (swr restored there)
19183
19184                                 ; --------------  SUBROUTINES AREA  -----------------
19185                                 ; the following subroutine causes a delay of a certain no of milliseconds
19186                                 ; the number of ms. delayed is based on the type of EEROM as indicated by R5
19187
19188 116754  010046                  DELAY:  MOV     R0,-(SP)                ; save R0
19189 116756  012700  023420                  MOV     #10000., R0             ; ALL ELSE = 10 MS DELAY
19190 116762  077001                  1$:     SOB     R0, 1$                  ; actual delay
19191 116764  012600                          MOV     (SP)+, R0               ; restore R0
19192 116766  000207                          RTS     PC
19193
19194                                 ;-------------------------------------------------------
19195 116770  000240                  TSTEND: NOP                             ; everything done
19196
```

```
 19198                                  .SBTTL  TEST  - LKS BIT 7
 19199                                  ; This test restores the software switch register used in the EEROM
 19200                                  ; test above, and therefore must follow it directly !! <<<<<<<<<*****
 19201                                  ;THESE TESTS WILL CHECK FUNCTIONALITY OF LKS<7-6>
 19202                                  ;
 19203                                  ;ACCESS LKS, CHECK THAT READY LKS<07> CAN BE 0 AND 1.
 19204                                  ;
 19205                                  ;LKS     <07>    LINE CLOCK MONITOR
 19206                                  ;        <06>    LINE CLOCK INTERRUPT ENABLE
 19207                                  ;ROUTINE TEST
 19208                                  ;IF UFD AND LKS IS DISABLED THEN
 19209                                  ;.       EXIT TEST
 19210                                  ;ENDIF
 19211                                  ;.       READ LKS TO SEE IF IT TIMEOUTS
 19212                                  ;.(CHECK LKS<07>)
 19213                                  ;.       LET R1=#77777(WORST CASE COUNTER FOR SLOW CLOCK)
 19214                                  ;.       LET 100=#ADDRESS OF LINE_CLOCK_INTERRUPT
 19215                                  ;.       CLEAR INTERRUPT_FLAG
 19216                                  ;.       DO 3 TIMES
 19217                                  ;.       .       REPEAT
 19218                                  ;.       .       .       DECREMENT R1
 19219                                  ;.       .       UNTIL R1 NE #0 OR LOW BYTE OF LKS LT #0
 19220                                  ;.       .       IF LOW BYTE OF LKS GE #0 THEN (READY DIDN'T COME UP)
 19221                                  ;.       .       .       ERROR
 19222                                  ;.       .       ENDIF
 19223                                  ;.       ENDDO
 19224                                  ;.       IF INTERRUPT_FLAG NE #0 THEN (INTERUPT W/O LKS<6>=1)
 19225                                  ;.       .       ERROR
 19226                                  ;.       ENDIF
 19227                                  ;ENDROUTINE
 19228
 19229                                  ;;*****************************************************************
     116772  000004                   TST35:  SCOPE
 19230 116774  013777  003050  062136          MOV     SAVSWR,@SWR                     ; restore here FROM EEROM test
 19231 117002  032737  000100  000052          BIT     #BIT06,@#52                     ;UFD MODE?
 19232 117010  001404                          BEQ     1$                              ;IF NOT, GO DO TEST
 19233 117012  032737  010000  177520          BIT     #BIT12,BCSR                     ;LKS DISABLED?
 19234 117020  001056                          BNE     TST36           ;;IF DISABLED, EXIT TEST
 19235                                  ;
 19236                                  ; TRY TO ACCESS LKS WITHOUT TIMEOUT
 19237                                  ;
 19238 117022  013737  000004  001160  1$:     MOV     ERRVEC,$TMP0                    ;SAVE TIMEOUT VECTOR
 19239 117030  012737  117052  000004          MOV     #2$,ERRVEC                      ;POINT NEW VECTOR TO PROGRAM
 19240 117036  012737  000340  000006          MOV     #340,ERRVEC+2                   ;AT PRIORITY 7
 19241 117044  005737  177546                  TST     LKS                             ;ACCESS LKS
 19242 117050  000403                          BR      3$                              ;IF NO TIMEOUT, CONTINUE
 19243 117052  104056                  2$:     ERROR   +56                             ;TIMEOUT READING LKS
 19244 117054  005726                          TST     (SP)+                           ;ADJUST STACK POINTER
 19245 117056  005726                          TST     (SP)+
 19246 117060  013737  001160  000004  3$:     MOV     $TMP0,ERRVEC                    ;RESTORE TIMEOUT VECTOR
 19247                                  ;
 19248                                  ; CHECK THAT READY BIT LKS<7> CAN BE 1
 19249                                  ;
 19250 117066  042737  000200  177546          BIC     #BIT07,LKS                      ; START WITH CLEAN SLATE
 19251 117074  005037  002722                  CLR     LKSFL                           ;CLEAR INTERRUPT FLAG
 19252 117100  012737  137062  000100          MOV     #LKSINT,100                     ;POINT VECTOR TO INERRUPT ROUTINE
 19253 117106  012737  000340  000102          MOV     #340,102                        ;AT PRIORITY 7
```

SEQ 0352

TEST - LKS BIT 7

```
 19254 117114  012702  000003            MOV     #3,R2                           ;DO 3 TIMES TO SYNCHRONISE
 19255 117120  012701  077777      4$:   MOV     #77777,R1                       ;COUNTER FOR SLOW CLOCKS
 19256 117124  105737  177546      5$:   TSTB    LKS                             ;READY LKS<7>=1?
 19257 117130  100401                    BMI     6$                              ;IF SO, DO NEXT LOOP
 19258 117132  077104                    SOB     R1,5$                           ;OTHERWISE, GO THRU COUNT
 19259 117134  105737  177546      6$:   TSTB    LKS                             ;WAS READY 1?
 19260 117140  100401                    BMI     7$                              ;IF YES, BRANCH
 19261 117142  104057                    ERROR   +57                             ;LKS<07> DOES NOT BECOME 1
 19262 117144  077213                7$: SOB     R2,4$                           ;DO ALL 3 TIMES
 19263 117146  005737  002722      8$:   TST     LKSFL                           ;ANY INTERRUPTS W/O LKS<6>=1?
 19264 117152  001401                    BEQ     TST36           ;;IF NONE, EXIT TEST
 19265 117154  104061                    ERROR   +61                             ;ILLEGAL CLOCK INTERRUPTS
 19266
```

```
19268                                   .SBTTL  TEST  - LKS INTERRUPT PRIORITY
19269                                   ;CHECK THAT LKS INTERRUPTS HAPPEN AT PRIORITY 5 CLEARING LKS<07>
19270                                   ;AND DON'T HAPPEN AT PRIORITY 6.
19271                                   ;ROUTINE TEST
19272                                   ;IF UFD AND LKS IS DISABLED THEN
19273                                   ;.      EXIT TEST
19274                                   ;ENDIF
19275                                   ;.      SET PRIORITY TO 5
19276                                   ;.      CLEAR INTERRUPT_FLAG
19277                                   ;.      LET LKS<06>=#1 (ENABLE INTERRUPTS)
19278                                   ;.      SET COUNTER TO WAIT FOR 3 INTERRUPTS
19279                                   ;.      REPEAT
19280                                   ;.      .       DECREMENT COUNTER
19281                                   ;.      UNTIL INTERRUPT_FLAG EQ #3 OR COUNTER EQ #0
19282                                   ;.      CLEAR LKS<06>
19283                                   ;.      IF LKS<07> EQ #1 THEN
19284                                   ;.      .       ERROR (WAS NOT CLEARED ON INTERRUPT)
19285                                   ;.      ENDIF
19286                                   ;.      IF COUNTER LT TIME_REQUIRED_FOR_3_INTERRUPTS_FOR_800HZ
19287                                   ;.      .       ERROR (INTERRUPTS NEVER GO LOW)
19288                                   ;.      ENDIF
19289                                   ;.      IF INTERRUPT_FLAG LT #3 THEN
19290                                   ;.      .       ERROR (INTERRUPTS DON'T HAPPEN)
19291                                   ;.      ENDIF
19292                                   ;.      CLEAR INTERUPT_FLAG
19293                                   ;.      WAIT FOR LKS<7>=1
19294                                   ;.      LET LKS<7>=0
19295                                   ;.      IF LKS<7> NE #0 THEN
19296                                   ;.      .       ERROR (LKS<7> NOT CLEARED
19297                                   ;.      ENDIF
19298                                   ;.      SET PRIORITY TO 6
19299                                   ;.      SET COUNTER TO 1 SLOW CLOCK INTERRUPT
19300                                   ;.      SET LKS<06>
19301                                   ;.      REPEAT
19302                                   ;.      .       DECREMENT COUNTER
19303                                   ;.      UNTIL COUNTER EQ #0 OR INTERRUPT_FLAG NE #0
19304                                   ;.      IF INTERRUPT_FLAG NE #0 THEN
19305                                   ;.      .       ERROR (INTERRUPT WAS AT WRONG PRIORITY)
19306                                   ;.      ENDIF
19307                                   ;.      RESTORE ORIGINAL PRIORITY
19308                                   ;ENDROUTINE
19309                                   ;
19310                                   ;ROUTINE LINE_CLOCK_INTERUPT
19311                                   ;.      INCREMENT INTERRUPT_FLAG
19312                                   ;ENDROUTINE
19313
19314                                   ;;*****************************************************************
    117156  000004                      TST36:  SCOPE
19315 117160  032737  000100  000052            BIT     #BIT06,@#52                     ;UFD MODE?
19316 117166  001404                            BEQ     1$                              ;IF NOT, GO DO TEST
19317 117170  032737  010000  177520            BIT     #BIT12,BCSR                     ;LKS DISABLED?
19318 117176  001132                            BNE     TST37           ;;IF DISABLED, EXIT TEST
19319                                   ;
19320                                   ; WAIT FOR 3 INTERRUPTS AND CHECK LKS<7> TO BE 0 AFTER INTERRUPT
19321                                   ;
19322 117200  042737  000100  177546  1$:     BIC     #BIT06,LKS                      ; FROM END OF TEST 42?? PROBLEM
19323 117206  005037  002722                    CLR     LKSFL                           ;CLEAR INTERRUPT FLAG
```

TEST - LKS INTERRUPT PRIORITY

```
19324 117212  012737  137062  000100          MOV     #LKSINT,100             ;POINT VECTOR TO ROUTINE
19325 117220  012701  077777                  MOV     #77777,R1               ;COUNTER FOR SLOW CLOCK
19326 117224  052737  000100  177546          BIS     #BIT06,LKS              ;SET INTERRUPT ENABLE BIT
19327 117232  032737  000100  177546          BIT     #BIT06,LKS              ;BIT SET OK?
19328 117240  001001                          BNE     2$                      ;IF YES, BRANCH
19329 117242  104131                          ERROR   +131                    ;ERROR WRITING 1 TO LKS<6>
19330 117244  106427  000240          2$:     MTPS    #240                    ;SET PRIORITY TO 5
19331 117250  022737  000003  002722  3$:     CMP     #3,LKSFL                ;3 INTERRUPTS HAPPENED?
19332 117256  001401                          BEQ     4$                      ;IF YES, BRANCH
19333 117260  077105                          SOB     R1,3$                   ;STAY IN A LOOP
19334                                 ;
19335                                 ; DISABLE INTERRUPTS AND CHECK THAT PROPER CONDITIONS ARE MET
19336                                 ;
19337 117262  042737  001000  177520  4$:     BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
19338 117270  106427  000340                  MTPS    #340                    ;RAISE PRIORITY
19339 117274  042737  000100  177546          BIC     #BIT06,LKS              ;DISABLE INTERRUPTS
19340 117302  032737  000200  177546          BIT     #BIT07,LKS              ;LKS<7> CLEARED AFTER INTERRUPTS?
19341 117310  001401                          BEQ     5$                      ;IF 0, BRANCH
19342 117312  104062                          ERROR   +62                     ;INTERRUPTS DON'T CLEAR LKS<7>
19343 117314  105737  177546          5$:     TSTB    LKS                     ;LKS<7>=1?
19344 117320  100375                          BPL     5$                      ;IF NOT, WAIT
19345 117322  005037  177546                  CLR     LKS                     ;CLEAR LKS<7>
19346 117326  032737  000200  177546          BIT     #BIT07,LKS              ;LKS<7> CLEARED?
19347 117334  001401                          BEQ     6$                      ;IF YES, BRANCH
19348 117336  104060                          ERROR   +60                     ;LKS<7> NOT CLEARED ON WRITE
19349 117340  032737  000100  177546  6$:     BIT     #BIT06,LKS              ;LKS<6>=0?
19350 117346  001401                          BEQ     7$                      ;IF YES, BRANCH
19351 117350  104131                          ERROR   +131                    ;ERROR WRITING 0 TO LKS<6>
19352 117352  052737  001000  177520  7$:     BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
19353 117360  022701  077737                  CMP     #77737,R1               ;COUNTER AT LESS THAN 800HZ?
19354 117364  002001                          BGE     8$                      ;IF NOT, BRANCH
19355 117366  104063                          ERROR   +63                     ;READY LINE DOES NOT GO LOW
19356 117370  022737  000003  002722  8$:     CMP     #3,LKSFL                ;DID 3 INTERRUPTS HAPPEN?
19357 117376  001404                          BEQ     9$                      ;IF YES, BRANCH
19358 117400  012737  000003  001124          MOV     #3,$GDDAT               ;3 INTERRUPTS EXPECTED
19359 117406  104064                          ERROR   +64                     ;INTERRUPTS DON'T HAPPEN
19360                                 ;
19361                                 ; CHECK WHETHER INTERRUPTS HAPPEN AT PRIORITY 6
19362                                 ;
19363 117410  005037  002722          9$:     CLR     LKSFL                   ;CLEAR INTERRUPT FLAG
19364 117414  106427  000300                  MTPS    #300                    ;RAISE PRIORITY TO 6
19365 117420  012701  077777                  MOV     #77777,R1               ;COUNTER FOR SLOW CLOCK
19366 117424  052737  000100  177546          BIS     #BIT06,LKS              ;SET INTERRUPT ENABLE BIT
19367 117432  005737  002722          10$:    TST     LKSFL                   ;ANY INTERRUPTS?
19368 117436  001001                          BNE     11$                     ;IF YES, EXIT LOOP
19369 117440  077104                          SOB     R1,10$                  ;CONTINUE WITH COUNT
19370 117442  005737  002722          11$:    TST     LKSFL                   ;ANY INTERRUPTS?
19371 117446  001404                          BEQ     12$                     ;IF NO, BRANCH
19372 117450  012737  000005  001124          MOV     #5,$GDDAT               ;STORE PRIORITY FOR TYPE OUT
19373 117456  104065                          ERROR   +65                     ;INTERUPTS HAPPEN AT WRONG PRIORITY
19374 117460  106427  000340          12$:    MTPS    #340                    ;RESTORE PRIORITY
19375
```

```
                                        .SBTTL  TEST  - LINE CLOCK DISABLE

                                        ;LINE CLOCK DISABLE(*)
                                        ;THIS TEST WILL CHECK THAT BCSR<12> DISABLES RESPONSE
                                        ;OF LKS REGISTER.
                                        ;
                                        ;BCSR   <12>    LINE CLOCK STATUS REGISTER DISABLE
                                        ;
                                        ;
                                        ;ROUTINE TEST
                                        ;
                                        ;IF UFD AND LKS IS NOT DISABLED THEN
                                        ;.      EXIT TEST
                                        ;ENDIF
                                        ;
                                        ;.      WRITE BCSR<12>=1
                                        ;.      LET 4=ADDRESS OF LKS_TRAP
                                        ;.      LET TRAP_LKS=0
                                        ;.      READ LKS
                                        ;.      IF TRAP_LKS NE 1 THEN
                                        ;.      .       ERROR
                                        ;.      ENDIF
                                        ;
                                        ;ENDROUTINE
                                        ;
                                        ;
                                        ;ROUTINE LKS_TRAP
                                        ;
                                        ;.      LET TRAP_LKS=1
                                        ;.      LET BCSR<12>=0
                                        ;
                                        ;RTI
                                        ;

                                        ;;*****************************************************************
 117464  000004                         TST37:  SCOPE
117466  032737  000100  000052                  BIT     #BIT06,@#52                     ;UFD MODE?
117474  001404                                  BEQ     1$                              ;IF NOT, BRANCH
117476  032737  010000  177520                  BIT     #BIT12,BCSR                     ;LKS DISABLED?
117504  001052                                  BNE     TST40           ;;IF DISABLED, EXIT TEST
                                        ;
                                        ; CHECK BCSR<12> TO BE 0 AND 1
                                        ;
117506  013737  177520  002730          1$:     MOV     BCSR,SAVBR                      ;SAVE BCSR REGISTER
117514  042737  010000  177520                  BIC     #BIT12,BCSR                             ;CLEAR BCSR
117522  032737  010000  177520                  BIT     #BIT12,BCSR                     ;<12>=0?
117530  001403                                  BEQ     2$                              ;IF OK, BRANCH
117532  005037  001124                          CLR     $GDDAT                          ;CLEAR EXPECTED PATTERN
117536  104052                                  ERROR   +52                             ;ERROR BCSR READ/WRITE BITS
117540  052737  010000  177520          2$:     BIS     #BIT12,BCSR                     ;SET BIT 12
117546  032737  010000  177520                  BIT     #BIT12,BCSR                     ;GOT SET OK?
117554  001004                                  BNE     3$                              ;IF OK, BRANCH
117556  012737  010000  001124                  MOV     #BIT12,$GDDAT                   ;EXPECTED PATTERN
117564  104052                                  ERROR   +52                             ;ERROR BCSR READ/WRITE BITS
                                        ;
                                        ; TRY TO ACCESS LKS TO GET A TIMEOUT WITH BCSR<12>=1
                                        ;
```

TEST - LINE CLOCK DISABLE

```
 19433 117566  013701  000004          3$:     MOV     ERRVEC,R1               ;SAVE TIMEOUT VECTOR
 19434 117572  012737  117614  000004          MOV     #4$,ERRVEC              ;POINT TO PROGRAM AREA
 19435 117600  012737  000340  000006          MOV     #340,ERRVEC+2           ;AT PRIORITY 7
 19436 117606  005737  177546                  TST     LKS                     ;ACCESS LKS REGISTER
 19437 117612  104066                          ERROR   +66                     ;BCSR<12> DOES NOT DISABLE LKS
 19438 117614  005726                  4$:     TST     (SP)+                   ;RESTORE STACK POINTER
 19439 117616  005726                          TST     (SP)+
 19440 117620  010137  000004                  MOV     R1,ERRVEC               ;RESTORE TIMEOUT VECTOR
 19441 117624  013737  002730  177520          MOV     SAVBR,BCSR              ;RESTORE BCSR
 19442
 19443
```

```
 19445                                  .SBTTL  TEST  - UNCONDITIONAL CLOCK LINE INTERRUPTS
 19446                                  ;UNCONDITIONAL CLOCK LINE INTERRUPTS(*)
 19447                                  ;THIS TEST WILL CHECK THAT SETTING BCSR<13> TO 1 WILL
 19448                                  ;REQUEST INTERRUPTS WHENEVER A CLOCK LINE IS ASSERTED.  THIS
 19449                                  ;SHOULD HAPPEN WITHOUT ACCESSING LKS, THEREFORE, EVEN WITH BCSR<12>=1
 19450                                  ;INTERRUPTS SHOULD HAPPEN.
 19451                                  ;
 19452                                  ;BCSR   <13>    FORCE LINE CLOCK INTERRUPT ENABLE
 19453                                  ;
 19454
 19455                                  ;ROUTINE TEST
 19456                                  ;IF UFD AND FORCE LKS NOT DISABLED THEN
 19457                                  ;.      EXIT TEST
 19458                                  ;ENDIF
 19459                                  ;.      LET 100=ADDRESS OF UNCONDITIONAL_INTERRUPT_ROUTINE
 19460                                  ;.      DO FOR BCSR<12> FROM #0 TO #1(LKS DISABLED AND ENABLED)
 19461                                  ;.      (IF UFD DO ONLY FOR SELECTED LINE CLOCK)
 19462                                  ;.      .       CLEAR UNCONDITIONAL_INTERRUPT
 19463                                  ;.      .       SET COUNTER TO WAIT  FOR 3 INTERRUPTS
 19464                                  ;.      .       LET BCSR<13>=1
 19465                                  ;.      .       REPEAT
 19466                                  ;.      .       .       DECREMENT COUNTER
 19467                                  ;.      .       UNTIL UNCONDITIONAL_INTERUPT EQ #3 OR COUNTER EQ #0
 19468                                  ;.      .       IF COUNTER GT TIME_REQUIRED_FOR_3_INTERRUPTS_FOR_800HZ
 19469                                  ;.      .       .       ERROR (INTERRUPTS NEVER GO LOW)
 19470                                  ;.      .       ENDIF
 19471                                  ;.      .       IF UNCONDITIONAL_INTERRUPT LT #3 THEN
 19472                                  ;.      .       .       ERROR (INTERRUPTS DON'T HAPPEN)
 19473                                  ;.      .       ENDIF
 19474                                  ;.      LET BCSR<13>=#0
 19475                                  ;.      ENDDO
 19476                                  ;ENDROUTINE
 19477                                  ;
 19478                                  ;ROUTINE UNCONDITIONAL_INTERRUPT_ROUTINE
 19479                                  ;.      INCREMENT UNCONDITIONAL_INTERRUPT
 19480                                  ;RETURN
 19481
 19482                                  ;;*****************************************************************
     117632  000004                     TST40:  SCOPE
 19483 117634  032737  000100  000052          BIT     #BIT06,@#52                     ;UFD MODE?
 19484 117642  001404                          BEQ     1$                              ;IF NOT, BRANCH
 19485 117644  032737  020000  177520          BIT     #BIT13,BCSR                     ;FORCE INTERRUPT SET?
 19486 117652  001530                          BEQ     TST41           ;;IF SET, EXIT TEST
 19487                                  ;
 19488                                  ; CHECK BSCR<13> TO BE 0 AND 1
 19489                                  ;
 19490 117654  013737  177520  002730  1$:     MOV     BCSR,SAVBR                      ;SAVE BCSR REGISTER
 19491 117662  042737  001000  177520          BIC     #1000,BCSR                      ;DISABLE HALT ON BREAK
 19492 117670  042737  020000  177520          BIC     #BIT13,BCSR                     ;CLEAR BCSR
 19493 117676  032737  020000  177520          BIT     #BIT13,BCSR                     ;<13>=0?
 19494 117704  001403                          BEQ     2$                              ;IF OK, BRANCH
 19495 117706  005037  001124                  CLR     $GDDAT                          ;CLEAR EXPECTED PATTERN
 19496 117712  104052                          ERROR   +52                             ;ERROR BCSR READ/WRITE BITS
 19497 117714  052737  020000  177520  2$:     BIS     #BIT13,BCSR                     ;SET BIT 13
 19498 117722  032737  020000  177520          BIT     #BIT13,BCSR                     ;GOT SET OK?
 19499 117730  001004                          BNE     3$                              ;IF OK, BRANCH
 19500 117732  012737  020000  001124          MOV     #BIT13,$GDDAT                   ;EXPECTED PATTERN
```

TEST - UNCONDITIONAL CLOCK LINE INTERRUPTS

```
19501 117740  104052                          ERROR   +52                     ;ERROR BCSR READ/WRITE BITS
19502                                ;
19503                                ; SET UP  TO DO UNCONDITIONAL INTERRUPTS
19504                                ;
19505 117742  012737  137062  000100  3$:     MOV     #LKSINT,@#100           ;SET UP INTERRUPT VECTOR
19506 117750  012737  000340  000102          MOV     #340,@#102              ;AT PRIORITY 7
19507 117756  052737  010000  177520          BIS     #BIT12,BCSR             ;FOR 1ST TIME DISABLE LKS
19508 117764  000403                          BR      5$                      ;GO DO IT
19509 117766  042737  010000  177520  4$:     BIC     #BIT12,BCSR             ;FOR THE 2ND TIME, ENABLE LKS
19510 117774  005037  002722          5$:     CLR     LKSFL                   ;CLEAR INTERRUPTS FLAG
19511 120000  012702  077777                  MOV     #77777,R2               ;COUNTER TO WAIT FOR INTERRUPTS
19512 120004  106427  000240                  MTPS    #240                    ;LOWER PRIORITY TO 5
19513 120010  022737  000003  002722  6$:     CMP     #3,LKSFL                ;3 INTERRUPTS HAPPENED?
19514 120016  001401                          BEQ     7$                      ;EXIT LOOP, IF SO
19515 120020  077205                          SOB     R2,6$                   ;OTHERWISE, KEEP WAITING
19516 120022  106427  000340          7$:     MTPS    #340                    ;RAISE PRIORITY TO 7
19517 120026  022702  077700                  CMP     #77700,R2               ;INTERRUPTS HAPPEN TOO OFTEN?
19518 120032  002001                          BGE     8$                      ;IF NOT, BRANCH
19519 120034  104063                          ERROR   +63                     ;READY LINE DOESN'T GO LOW
19520 120036  022737  000003  002722  8$:     CMP     #3,LKSFL                ;AT LEAST 3 INTERRUPTS HAPPENED?
19521 120044  002004                          BGE     9$                      ;IF SO, BRANCH
19522 120046  012737  000003  00112           MOV     #3,$GDDAT               ;EXPECTED DATA
19523 120054  104064                          ERROR   +64                     ;INTERRUPTS DON'T HAPPEN
19524 120056  032737  010000  177520  9$:     BIT     #BIT12,BCSR             ;SECOND TIME THRU THE LOOP?
19525 120064  001340                          BNE     4$                      ;IF NOT, DO IT AGAIN
19526 120066  032737  000100  000052          BIT     #BIT06,@#52             ;UFD MODE?
19527 120074  001404                          BEQ     10$                     ;IF NOT, BRANCH
19528 120076  032737  010000  177520          BIT     #BIT12,BCSR             ;IF UFD AND LKS DISABLED?
19529 120104  001010                          BNE     12$                     ;DON'T CHECK LKS
19530 120106  032737  000100  177546  10$:    BIT     #BIT06,LKS              ;INTERRUPT ENABLE LINE HOLD 1?
19531 120114  001001                          BNE     11$                     ;IF SO, BRANCH
19532 120116  104067                          ERROR   +67                     ;BCSR<13> DOESN'T SET LKS<6>
19533 120120  042737  000100  177546  11$:    BIC     #BIT06,LKS              ;DISABLE LKS INTERRUPTS
19534 120126  013737  002730  177520  12$:    MOV     SAVBR,BCSR              ;RESTORE BCSR
19535
19536
```

```
                                .SBTTL  TEST  - RESETTING LKS
                                ;RESETTING LKS(*)
                                ;THIS TEST WILL PROVE THAT RESET INSTRUCTION SETS LKS<07> AND
                                ;CLEARS LKS<06>.
                                ;ROUTINE TEST
                                ;IF UFD AND LKS IS DISABLED THEN
                                ;.      EXIT TEST
                                ;ENDIF
                                ;.      POINT LKS VECTOR 100 TO ERROR_LKS_ILLEGAL_INTERRUPT
                                ;.      SYNCHRONIZE LKS BY WAITING FOR 3 PULSES
                                ;.      LET LKS<06>=#1
                                ;.      CLEAR LKS (CLEARS LKS<07>
                                ;.      EXECUTE "RESET"
                                ;.      IF LKS<7> NE #1 OR LKS<6> NE #0 THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;.      IF ILLEGAL_LINE_CLOCK_INTERRUPT NE 0 THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;ENDROUTINE
                                ;
                                ;ROUTINE ERROR_LKS_ILLEGAL_INTERRUPT
                                ;.      FLAG ILLEGAL_LINE_CLOCK_INTERRUPT
                                ;RETURN

                                ;;*****************************************************************
                                TST41:  SCOPE
 120134  000004
19564 120136  005737  001206          TST     $PASS                           ;FIRST PASS?
19565 120142  001057                  BNE     TST42           ;;IF NOT FIRST PASS, EXIT TEST
19566 120144  032737  000100  000052  BIT     #BIT06,@#52                     ;UFD MODE?
19567 120152  001404                  BEQ     1$                              ;IF NOT, BRANCH
19568 120154  032737  010000  177520  BIT     #BIT12,BCSR                     ;LKS DISABLED?
19569 120162  001447                  BEQ     TST42           ;;IF DISABLED, EXIT TEST
19570                           ;
19571                           ; SYNCHRONISE WITH LINE TIME CLOCK BY WAITING FOR 3 INTERRUPTS
19572                           ;
19573 120164  013737  177520  002730  1$:     MOV     BCSR,SAVBR                      ;SAVE BCSR
19574 120172  042737  010000  177520        BIC     #BIT12,BCSR                     ;ENABLE LKS RESPONSE
19575 120200  012737  137062  000100        MOV     #LKSINT,@#100                   ;SET UP INTERRUPT VECTOR
19576 120206  012737  000340  000102        MOV     #340,@#102                      ;AT PROIRITY 7
19577 120214  052737  000100  177546        BIS     #BIT06,LKS                      ;SET INTERRUPT ENABLE BIT
19578 120222  005037  002722                CLR     LKSFL                           ;CLEAR INTERRUPTS FLAG
19579 120226  012702  077777                MOV     #77777,R2                       ;COUNTER TO WAIT FOR INTERRUPTS
19580 120232  106427  000240                MTPS    #240                            ;LOWER PRIORITY TO 5
19581 120236  022737  000003  002722  2$:     CMP     #3,LKSFL                        ;3 INTERRUPTS HAPPENED?
19582 120244  001401                        BEQ     3$                              ;EXIT LOOP, IF SO
19583 120246  077205                        SOB     R2,2$                           ;OTHERWISE, KEEP WAITING
19584 120250  106427  000340          3$:     MTPS    #340                            ;RAISE PRIORITY TO 7
19585 120254  000005                        RESET                                   ;EXECUTE RESET
19586 120256  032737  000200  177546        BIT     #BIT07,LKS                      ;READY BIT SET?
19587 120264  001001                        BNE     4$                              ;IF SO, BRANCH
19588 120266  104070                        ERROR   +70                             ;RESET DOESN'T SET LKS<07.
19589 120270  032737  000100  177546  4$:     BIT     #BIT06,LKS                      ;INTERRUPT ENABLE BIT CLEARED?
19590 120276  001401                        BEQ     TST42           ;;IF SO, EXIT TEST
19591 120300  104071                        ERROR   +71                             ;RESET DOESN'T CLEAR LKS
19592
19593
```

```
 19595                                  .SBTTL  TEST  - LINE CLOCK INTERRUPTS
 19596                                  ;LINE CLOCK INTERRUPTS(*)
 19597                                  ;BY SETTING TO 1 LKS<06>, THIS TEST WILL CHECK FOR INTERRUPTS
 19598                                  ;FROM BEVENT LINE AND FROM KDJ11-B 50HZ, 60HZ, 800HZ ON BOARD SIGNALS
 19599                                  ;(THE LATTER SIGNALS WILL BE ACCESSED BY SETTING BCSR<11-10>).
 19600                                  ;
 19601                                  ;BCSR   <11>    <10>            CLOCK SELECT BITS 1 AND 0
 19602                                  ;
 19603                                  ;        0       0              EXTERNAL BEVENT LINE
 19604                                  ;        0       1              ON-BOARD 50 HZ
 19605                                  ;        1       0              ON-BOARD 60 HZ
 19606                                  ;        1       1              ON-BOARD 800 HZ
 19607                                  ;
 19608                                  ;ROUTINE TEST
 19609                                  ;IF UFD THEN
 19610                                  ;.      IF LKS DISABLED THEN
 19611                                  ;.      .       EXIT TEST
 19612                                  ;.      ENDIF
 19613                                  ;.      SET FLAGS TO RUN ONLY WHAT SPECIFIED IN EAPROM
 19614                                  ;ENDIF
 19615                                  ;.      LET 100=ADDRESS OF LKS_INTERRUPT
 19616                                  ;.      DO FOR BCSR<11;10> FROM #0 TO #3
 19617                                  ;.      .       LET LKS<06>=#1
 19618                                  ;.      .       WAIT FOR 10 INTERRUPTS FOR EACH CLOCK
 19619                                  ;.      .       STORE ACTUAL NUMBER OF INTERRUPTS FOR EACH CLOCK
 19620                                  ;.      .       LET INTERRUPT_FLAG=0
 19621                                  ;.      ENDDO
 19622                                  ;.      COMPARE NUMBER OF INTERRUPTS FOR EACH CLOCK
 19623                                  ;ENDROUTINE
 19624                                  ;
 19625                                  ;ROUTINE LKS_INTERRUPT
 19626                                  ;.      INCREMENT INTERRUPT_FLAG
 19627                                  ;RETURN
 19628
 19629                                  ;;*****************************************************************
      120302  000004                   TST42:  SCOPE
 19630 120304  000240                           NOP                             ; THIS CODE ADDED FOR APT DEFAULT BREAK
 19631                                                                          ; PURPOSES-- TEST TIME LONGER THAN SOME APT BREAK IN
TERVALS !
 19632 120306  005737  003032                   TST     CCHPAS                  ;have done enough inclusive passes?
 19633 120312  001003                           BNE     99$                     ; not yet
 19634 120314  000240                           NOP                             ; debug aid
 19635 120316  000137  120564                   JMP     10$                     ; yes skip this
 19636 120322  000240                   99$:    NOP
 19637 120324  032737  000100  000052           BIT     #BIT06,@#52                     ;UFD MODE?
 19638 120332  001411                           BEQ     1$                              ;IF NOT, GO DO THE TEST
 19639 120334  032737  010000  177520           BIT     #BIT12,BCSR                     ;LKS IS DISABLED?
 19640 120342  001123                           BNE     TST43           ;;IF DISABLED, EXIT TESTS
 19641 120344  005002                           CLR     R2                              ;CLEAR R2 TO SET FLAGS
 19642 120346  053702  177520                   BIS     BCSR,R2                         ;SET R2 ACCORDING TO BCSR
 19643 120352  042702  171777                   BIC     #171777,R2                      ;LEAVE ONLY BITS <11-10>
 19644                                  ;
 19645                                  ; SETUP DELAY VALUES FOR INTERRUPTS, IN UFD MODE ONLY FROM THE CLOCK SPECIFIED IN BCSR<11-10
>
 19646                                  ;
 19647 120356  013737  177520  002730   1$:     MOV     BCSR,SAVBR                      ;STORE BCSR
 19648 120364  012737  137062  000100           MOV     #LKSINT,100                     ;SET UP LKS VECTOR
 19649 120372  012737  000340  000102           MOV     #340,102                        ;AT PRIORITY 7
 19650 120400  012705  120602                   MOV     #TIMDEL,R5                      ;POINTER TO DEL
```

TEST - LINE CLOCK INTERRUPTS

```
19651 120404  012704  000004                    MOV     #4,R4                   ;R4 IS THE COUNTER FOR ALL CLOCKS
19652 120410  005037  177520                    CLR     BCSR                    ;DO FOR BEVENT LINE INTERRUPTS
19653 120414  000403                            BR      3$                      ;GO DO THE LOOP
19654 120416  062737  002000  177520  2$:       ADD     #2000,BCSR              ;SET UP FOR THE NEXT CLOCK LINE
19655 120424  032737  000100  000052  3$:       BIT     #BIT06,@#52             ;UFD MODE?
19656 120432  001402                            BEQ     4$                      ;IF NOT, BRANCH
19657 120434  010237  177520                    MOV     R2,BCSR                 ;IN UFD, DO ONLY FOR SPECIFIED
19658 120440  005037  002722            4$:     CLR     LKSFL                   ;CLEAR INTERRUPT FLAG
19659 120444  052737  000100  177546            BIS     #BIT06,LKS              ;SET INTERRUPT ENABLE BIT
19660 120452  012703  000010                    MOV     #10,R3                  ;START COUNTER FOR 10 INTERURRUPTS
19661 120456  012701  177777            5$:     MOV     #177777,R1              ;START COUNTER TO WAIT FOR INTERRUPT
19662 120462  106427  000240                    MTPS    #240                    ;LOWER PRIORITY TO 5
19663 120466  023703  002722            6$:     CMP     LKSFL,R3                ;NEW INTERRUPT HAPPENED?
19664 120472  001401                            BEQ     7$                      ;IF SO, EXIT WAIT LOOP
19665 120474  077104                            SOB     R1,6$                   ;OTHERWISE, KEEP WAITING
19666 120476  010125            7$:     MOV     R1,(R5)+                ;STORE DELAY FOR EACH CLOCK
19667 120500  032737  000100  000052            BIT     #BIT06,@#52             ;UFD MODE?
19668 120506  001026                            BNE     10$                     ;IF UFD, DON'T DO FOR ANY OTHER
19669 120510  077436                            SOB     R4,2$                   ;ALL LINE CLOCKS DONE?
19670                                   ;
19671                                   ; CHECK THE DELAY VALUES FOR ALL CLOCKS
19672                                   ;
19673 120512  106427  000340                    MTPS    #340                    ;RAISE PRIORITY
19674 120516  042737  000100  177546            BIC     #BIT06,LKS              ;DISABLE INTERRUPTS
19675 120524  012705  120602                    MOV     #TIMDEL,R5              ;POINTER TO DELAY TABLE
19676 120530  021565  000006                    CMP     (R5),6(R5)              ;DELAY FOR BEVENT AND 800HZ?
19677 120534  103401                            BLO     8$                      ;BEVENT IS NOT 800HZ
19678 120536  104064                            ERROR   +64                     ;WRONG # OF INTERRUPTS
19679 120540  005725            8$:     TST     (R5)+                   ;INCREMENT POINTER
19680 120542  021565  000002                    CMP     (R5),2(R5)              ;DELAY FOR 50HZ AND 60HZ?
19681 120546  103401                            BLO     9$                      ;IF FIRST BIGGER, BRANCH
19682 120550  104064                            ERROR   +64                     ;WRONG # OF INTERRUPTS
19683 120552  005725            9$:     TST     (R5)+                   ;INCREMENT POINTER
19684 120554  021565  000002                    CMP     (R5),2(R5)              ;DELAY FOR 50HZ AND 800HZ
19685 120560  103401                            BLO     10$                     ;IF FIRST BIGGER, BRANCH
19686 120562  104064                            ERROR   +64                     ;WRONG # OF INTERRUPTS
19687 120564  013737  002730  177520  10$:      MOV     SAVBR,BCSR              ;RESTORE BCSR
19688 120572  042737  000100  177546            BIC     #BIT06,LKS              ;DISABLE INTERRUPTS
19689 120600  000404                            BR      TST43           ;;EXIT TEST
19690
19691 120602                            TIMDEL: .BLKW 4
19692
```

TEST - MAINTENANCE REGISTER TEST

```
                                    .SBTTL  TEST  - MAINTENANCE REGISTER TEST
                                    ;MAINTENANCE REGISTER TEST
                                    ;THIS TEST WILL ADDRESS MAINTENANCE REGISTER AND CHECK BITS
                                    ;7-4 TO BE 0010, 2-1 TO BE 10, AND READ BITS 10-08, 03, 00
                                    ;FOR FUTURE USE.  THOSE BITS REPRESENT THE FOLLOWING SIGNALS:
                                    ;MULTIPROCESSOR SLAVE, UNIBUS SYSTEM, FPA AVAILABLE, HALT/TRAP
                                    ;OPTION, AND AC POWER OKAY.
                                    ;ROUTINE TEST
                                    ;.      IF MAINT. REG. BITS <7-4> NE 0010 OR <2-1> NE 10 THEN
                                    ;.      .       ERROR
                                    ;.      ENDIF
                                    ;.      READ MAINT.REG. BITS <10-08,03,00>
                                    ;ENDROUTINE

                                    ;;*****************************************************************
    120612  000004                  TST43:  SCOPE
19709 120614  032737  174000  177750        BIT     #174000,MAIREG                  ;UNUSED BITS ALL ZEROS?
19710 120622  001401                        BEQ     1$                              ;IF OK, BRANCH
19711 120624  104132                        ERROR   +132                            ;MAINTENANCE REGISTER ERROR
19712 120626  032737  000044  177750  1$:   BIT     #44,MAIREG                      ;<5,2> SET ?
19713 120634  001001                        BNE     2$                              ;IF SO, BRANCH
19714 120636  104132                        ERROR   +132                            ;MAINTENANCE REGISTER ERROR
19715 120640  032737  000322  177750  2$:   BIT     #322,MAIREG                     ;<7,6,4,1> CLEAR?
19716 120646  001401                        BEQ     TST44           ;;IF YES, BRANCH
19717 120650  104132                        ERROR   +132
19718
19719
```

```
                                .SBTTL  TEST  - SERIAL LINE UNIT REGISTERS
                                ;SERIAL LINE UNIT TEST(*)
                                ;BCR<2-0> WILL BE READ TO FIND OUT BAUD RATE. SLU WILL BE PROG-
                                ;RAMMED TO CHECK THE INTERRUPT LEVELS BY SETTING BIT<06> IN
                                ;RCSR AND XMIT. LOOP BACK CAPABILITIES WILL BE TESTED BY SETTING
                                ;TO 1 XCSR<02>.   THE LINE CLOCK INTERRUPT SUBROUTINE WILL BE
                                ;USED TO RETURN TO THE EXECUTION OF THE DIAGNOSTICS, IF THE
                                ;PROGRAM HANGS IN THE LOOP BACK MODE.
                                ;ROUTINE TEST
                                ;IF UFD AND CONSOLE NOT PRESENT
                                ;.      GO TO TEST_22
                                ;ENDIF
                                ;.      IF BCR<07> EQ #0 THEN
                                ;.      .       READ BCR<2-0> TO GET BAUD RATE
                                ;.      ENDIF
                                ;.      LET 4=ADDRESS_OF_TIMEOUT _ROUTINE
                                ;.      DO FOR RCSR,XCSR,RBUF,XBUF
                                ;.      .       READ XRCSR,XCSR,RBUF,XBUF
                                ;.      .       IF TIMEOUT_FLAG NE #0 THEN
                                ;.      .       .       ERROR
                                ;.      .       ENDIF
                                ;.      ENDDO
                                ;ENDROUTINE
                                ;
                                ;ROUTINE TIMEOUT
                                ;.      LET TIMEOUT_FLAG=#1
                                ;ENDROUTINE

                                ;;*****************************************************************
    120652  000004              TST44:  SCOPE
120654  032737  000100  000052          BIT     #BIT06,@#52             ;UFD MODE?
120662  001406                        BEQ     1$                      ;IF NOT, GO DO THE TEST
120664  032737  000200  177524        BIT     #BIT07,BCR              ;IF UFD AND CONSOLE NOT PRESENT
120672  001402                        BEQ     1$                      ;NOT TRUE, DO THE TEST
120674  000137  122620                JMP     SLEND                   ;IF TRUE, SKIP ALL SLU TESTS
                                ;
                                ; TRY TO ACCESS SLU REGISTERS
                                ;
120700  013701  000004          1$:     MOV     ERRVEC,R1               ;SAVE TIMEOUT VECTOR
120704  012737  120730  000004        MOV     #3$,ERRVEC              ;POINT NEW ONE TO PROGRAM AREA
120712  012737  000340  000006        MOV     #340,ERRVEC+2           ;AT PRIORITY 7
120720  012702  177560                MOV     #RCSR,R2                ;START ACCESSING WITH RCSR
120724  005712                  2$:     TST     (R2)                    ;ACCESS SLU REGISTER
120726  000403                        BR      4$                      ;IF NO TIMEOUT, CONTINUE
120730  010237  001126          3$:     MOV     R2,$BDDAT               ;STORE ADDRESS THAT TIMED OUT
120734  104072                        ERROR   .72                     ;TIMEOUT ACCESSING SLU REGISTER
120736  022722  177566          4$:     CMP     #XBUF,(R2)+             ;LAST REGISTER ACCESSED?
120742  103770                        BLO     2$                      ;IF NOT, BRANCH
120744  010137  000004                MOV     R1,ERRVEC               ;RESTORE TIMEOUT VECTOR
```

TEST - XCSR BIT 7

```
 19771                                  .SBTTL  TEST  - XCSR BIT 7
 19772                                  ;CHECK THAT XCSR<07> CAN BE 0 AND 1.
 19773                                  ;
 19774                                  ;XCSR   <07>            TRANSMITTER READY
 19775                                  ;
 19776                                  ;ROUTINE TEST
 19777                                  ;.      WAIT FOR XCSR<07>=#1 NO MORE THAN 200MSEC
 19778                                  ;.      IF XCSR<07> NE #1 THEN
 19779                                  ;.      .       ERROR
 19780                                  ;.      ENDIF
 19781                                  ;.      LET XBUF=#NULL
 19782                                  ;.      WAIT FOR XCSR<07>=#1
 19783                                  ;.      LET XBUF=#NULL
 19784                                  ;.      IF XCSR<07> NE 0 THEN
 19785                                  ;.      .       ERROR (READY DIDN'T GO LOW)
 19786                                  ;.      ENDIF
 19787                                  ;ENDROUTINE
 19788
 19789                                  ;;**********************************************************************
     120750  000004                     TST45:  SCOPE
 19790 120752  012701  001000                   MOV     #1000,R1                        ;COUNTER FOR ABOUT 200MICROSEC.
 19791 120756  122737  000001  001220           CMPB    #APTENV,$ENV                    ;RUNNING IN APT MODE?
 19792 120764  001003                           BNE     1$                              ;NO, GO DO TEST
 19793 120766  005737  001206                   TST     $PASS                           ;FIRST PASS?
 19794 120772  001017                           BNE     TST46           ;;IF APT AND NOT FIRST PASS, EXIT TEST
 19795 120774  105737  177564           1$:     TSTB    XCSR                            ;XCSR<7> READY 1?
 19796 121000  100401                           BMI     2$                              ;IF SO, EXIT WAIT LOOP
 19797 121002  077104                           SOB     R1,1$                           ;IF NOT 1, CONTINUE WAITING
 19798 121004  105737  177564           2$:     TSTB    XCSR                            ;XCSR<7>=1?
 19799 121010  100401                           BMI     3$                              ;IF YES, BRANCH
 19800 121012  104073                           ERROR   .73                             ;XCSR<7> DOES NOT BECOME 1
 19801 121014  012737  000000  177566   3$:     MOV     #NULL,XBUF                      ;TRY TO TRANSMIT NULL CHARACTER
 19802 121022  105737  177564                   TSTB    XCSR                            ;XCSR<7>=0
 19803 121026  100001                           BPL     TST46           ;;IF YES, EXIT TEST
 19804 121030  104073                           ERROR   .73                             ;XMIT READY DIDN'T GO LOW
 19805
 19806
```

TEST - RCSR BIT 7 AND XCSR BIT 2

```
                                .SBTTL  TEST  - RCSR BIT 7 AND XCSR BIT 2
                                ;CHECK THAT RCSR<07> CAN BE 0 AND 1 AND THAT XCSR<02> WORKS PROPERLY.
                                ;
                                ;RCSR   <07>    RECEIVER DONE
                                ;XCSR   <02>    MAINTENANCE
                                ;
                                ;ROUTINE TEST
                                ;.(CHECK RCSR<07> AND XCSR<07>)
                                ;.      WAIT FOR XCSR<07>=#1
                                ;.      LET XCSR<02>=#1 (LOOP BACK MODE)
                                ;.      LET XBUF=#125
                                ;.      WAIT FOR RCSR<07>=#1 NO MORE THAN 200MSEC
                                ;.      IF RCSR<07> NE #1 THEN
                                ;.      .       ERROR (RCSR<07> DOES NOT BECOME 1 OR XCSR<02>DOES NOT
                                ;.      .               WORK)
                                ;.      ENDIF
                                ;.      IF RBUF NE #125 THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;.      IF RCSR<07> NE #0 THEN
                                ;.      .       ERROR (RCSR<07>DOES NOT GO LOW)
                                ;.      ENDIF
                                ;.      LET XCSR<02>=#0
                                ;ENDROUITNE

                                ;;*****************************************************************
 121032  000004                 TST46:  SCOPE
19834 121034  012701  000013            MOV     #13,R1                  ;COUNTER FOR ABOUT 200MICROSEC.
19835 121040  122737  000001  001220      CMPB    #APTENV,$ENV            ;RUNNING IN APT MODE?
19836 121046  001003                      BNE     1$                      ;NO, GO DO TEST
19837 121050  005737  001206              TST     $PASS                   ;FIRST PASS?
19838 121054  001074                      BNE     TST47           ;;IF APT AND NOT FIRST PASS, EXIT TEST
19839 121056  105737  177564      1$:     TSTB    XCSR                    ;XCSR<7> READY 1?
19840 121062  100375                      BPL     1$                      ;IF NOT 1, CONTINUE WAITING
19841 121064  052737  000004  177564  2$: BIS     #BIT02,XCSR             ;SET LOOP BACK MODE
19842 121072  032737  000004  177564      BIT     #BIT02,XCSR             ;GOT SET OK?
19843 121100  001004                      BNE     3$                      ;IF YES, BRANCH
19844 121102  005037  177564              CLR     XCSR                    ;RESET TO PRINT ERROR
19845 121106  104114                      ERROR   +114                    ;XCSR<2> DOES NOT BECOME 1
19846 121110  000456                      BR      TST47           ;;EXIT TEST
19847                                   ;
19848                                   ; STALL FOR A WHILE IN CASE XCSR<2> CAUSES RCSR<7> TO BE 1
19849                                   ;
19850 121112  012701  060000      3$:     MOV     #60000,R1               ;STALL IN CASE XCSR<2> SETS READY
19851 121116  105737  177560      4$:     TSTB    RCSR                    ;IF RECEIVER READY SET?
19852 121122  100402                      BMI     5$                      ;IF SET, BRANCH
19853 121124  077104                      SOB     R1,4$                   ;OTHERWISE, STAY FOR A WHILE
19854 121126  000402                      BR      6$                      ;IF NOT READY, BRANCH
19855 121130  005737  177562      5$:     TST     RBUF                    ;READ RBUF
19856                                   ;
19857                                   ; TRANSMIT XON AND CHECK RCSR<7>
19858                                   ;
19859 121134  012737  000021  177566  6$: MOV     #21,XBUF                ;TRANSMIT A CHARACTER
19860 121142  012701  060000              MOV     #60000,R1               ;COUNTER TO WAIT
19861 121146  105737  177560      7$:     TSTB    RCSR                    ;RCSR<7> READY 1?
19862 121152  100401                      BMI     8$                      ;IF YES, EXIT WAIT LOOP
19863 121154  077104                      SOB     R1,7$                   ;OTHERWISE, CONTINUE WAITING
```

TEST - RCSR BIT 7 AND XCSR BIT 2

```
 19864 121156  105737  177560          8$:    TSTB    RCSR              ;RCSR<7>=1?
 19865 121162  100403                         BMI     9$                ;IF YES, BRANCH
 19866 121164  005037  177564                 CLR     XCSR              ;RESET XCSR<2>
 19867 121170  104074                         ERROR   ;74               ;RECEIVER READY DIDN'T COME UP
 19868 121172  013737  177562  001126  9$:    MOV     RBUF,$BDDAT       ;STORE RECEIVED DATA
 19869 121200  022737  000021  001126         CMP     #21,$BDDAT        ;DATA RECEIVED OK?
 19870 121206  001406                         BEQ     10$               ;IF YES, BRANCH
 19871 121210  012737  000021  001124         MOV     #21,$GDDAT
 19872 121216  005037  177564                 CLR     XCSR              ;RESET TO ENABLE SLU
 19873 121222  104075                         ERROR   ;75               ;WRONG CHARACTER RECEIVED
 19874 121224  105737  177560          10$:   TSTB    RCSR              ;RCSR<7>=0?
 19875 121230  100003                         BPL     11$               ;IF ZERO, BRANCH
 19876 121232  005037  177564                 CLR     XCSR              ;RESET TO ENABLE SLU
 19877 121236  104076                         ERROR   ;76               ;RCSR<07><>0 AFTER READING RBUF
 19878 121240  042737  000004  177564  11$:    BIC     #BIT02,XCSR       ;DISABLE LOOP BACK MODE
 19879
 19880
```

TEST - RESET AND XCSR<2!0>

```
19882                                   .SBTTL  TEST  - RESET AND XCSR<2!0>
19883                                   ;CHECK THAT RESET CLEARS XCSR<0!2>.
19884                                   ;ROUTINE TEST
19885                                   ;.(CHECK RCSR<07> AND XCSR<07> AND RESET)
19886                                   ;.      LET XCSR<02,00>=#1 (LOOP BACK MODE)
19887                                   ;.      EXECUTE "RESET"
19888                                   ;.      IF XCSR<02!00> NE #0 THEN
19889                                   ;.          ERROR
19890                                   ;.      ENDIF
19891                                   ;.      LET XCSR<02>=#0
19892                                   ;ENDROUTINE
19893
19894                                   ;;*****************************************************************
      121246  000004                    TST47:  SCOPE
19895 121250  005737  001206                    TST     $PASS                           ;FIRST PASS?
19896 121254  001011                            BNE     TST50           ;;IF NOT FIRST PASS, EXIT TEST
19897 121256  052737  000005  177564    1$:     BIS     #BIT02!BIT00,XCSR               ;LOOP BACK MODE
19898                                   ;
19899                                   ; EXECUTE RESET AND VALIDATE THAT XCSR<7,2> BECOMES <1,0>
19900                                   ;
19901 121264  000005                            RESET                                   ;EXECUTE RESET
19902 121266  032737  000005  177564            BIT     #BIT02!BIT00,XCSR               ;XCSR<2,0> CLEAR?
19903 121274  001401                            BEQ     TST50           ;;IF YES, BRANCH
19904 121276  104102                            ERROR   +102                            ;XCSR<2,0> NOT CLEARED ON RESET
19905
19906
```

```
 19908                                  .SBTTL  TEST  - RESET AND INTERRUPT ENABLE BITS
 19909                                  ;CHECK THAT INTERRUPTS DON'T HAPPEN AT PRIORITY 4 AND THAT RESET
 19910                                  ;CLEARS XCSR<06> AND RCSR<06>.
 19911                                  ;
 19912                                  ;RCSR   <06>    RECEIVER INTERRUPT ENABLE
 19913                                  ;XCSR   <06>    TRANSMITTER INTERRUPT ENABLE
 19914                                  ;
 19915                                  ;ROUTINE TEST
 19916                                  ;.      LET 60=#ADDRESS_OF_ILLEGAL_INTERRUPT_XRCSR
 19917                                  ;.      LET 64=#ADDRESS_OF_ILLEGAL_INTERRUPT_XRCSR
 19918                                  ;.      SET PRIORITY TO 4
 19919                                  ;.      LET XCSR<02>=#1 (LOOPBACK MODE)
 19920                                  ;.      LET XCSR<06>=#1 (ENABLE TRANSMIT INTERRUPTS)
 19921                                  ;.      LET RCSR<06>=#1 (ENABLE RECEIVE INTERRUPTS)
 19922                                  ;.      WAIT FOR XCSR<07>=#1 (READY TO TRANSMIT)
 19923                                  ;.      LET XBUF=#NULL (SEND A CHARACTER)
 19924                                  ;.      WAIT FOR ILLEGAL INTERRUPTS (ABOUT 200MSEC)
 19925                                  ;.      EXECUTE "RESET"
 19926                                  ;.      IF XCSR<06> NE #0 OR RCSR<06> NE #0 OR XRCSR NE #0 THEN
 19927                                  ;.              ERROR
 19928                                  ;.      ENDIF
 19929                                  ;.      RESTORE PRIORITY TO NORMAL
 19930                                  ;ENDROUTINE
 19931                                  ;
 19932                                  ;ROUTINE ILLEGAL_INTERRUPT_XRCSR
 19933                                  ;.      INCREMENT XRCSR
 19934                                  ;ENDROUTINE
 19935
 19936                                  ;;*****************************************************************
      121300  000004                    TST50:  SCOPE
 19937 121302  005737  001206                    TST     $PASS                   ;FIRST PASS?
 19938 121306  001033                    BNE     5$                      ;SKIP RESET PART OF THE TEST
 19939                                  ;
 19940                                  ; CHECK THAT INTERRUPTS ENABLE BITS FOR RECEIVER AND TRASMITTER OF SLU
 19941                                  ; ARE CLEARED BY RESET
 19942                                  ;
 19943 121310  052737  000100  177564  1$:     BIS     #BIT06,XCSR             ;SET INTERRUPT ENABLE BIT IN XCSR
 19944 121316  032737  000100  177564          BIT     #BIT06,XCSR             ;GOT SET OK?
 19945 121324  001001                    BNE     2$                      ;IF YES, BRANCH
 19946 121326  104110                    ERROR   +110                    ;IN BIT 6 OF XCSR
 19947 121330  052737  000100  177560  2$:     BIS     #BIT06,RCSR             ;SET INTERRUPT ENABLE BIT IN RCSR
 19948 121336  032737  000100  177560          BIT     #BIT06,RCSR             ;GOT SET OK?
 19949 121344  001001                    BNE     3$                      ;IF YES, BRANCH
 19950 121346  104110                    ERROR   +110                    ;IN BIT 6 OF RSCR
 19951 121350  000005            3$:     RESET                           ;INLINE BUS RESET
 19952 121352  032737  000100  177564          BIT     #BIT06,XCSR             ;XMIT INTERRUPT ENABLE BIT CLEARED?
 19953 121360  001401                    BEQ     4$                      ;IF CLEARED, BRANCH
 19954 121362  104102                    ERROR   +102                    ;INTERRUPT ENABLE NOT CLEARED ON RESET
 19955 121364  032737  000100  177560  4$:     BIT     #BIT06,RCSR             ;RECEIVE INTERRUPT ENBLE CLEARED?
 19956 121372  001401                    BEQ     5$                      ;IF CLEARED, BRANCH
 19957 121374  104102                    ERROR   +102                    ;INTERRUPT ENABLE NOT CLEARED ON RESET
 19958                                  ;
 19959                                  ; CHECK THAT TRANSMIT INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN 3
 19960                                  ;
 19961 121376  012737  121444  000064  5$:     MOV     #9$,@#64                ;POINT XMIT VECTOR TO PROGRAM AREA
 19962 121404  012737  000340  000066          MOV     #340,@#66               ;AT PRIORITY 7
 19963 121412  052737  000100  177564          BIS     #BIT06,XCSR             ;SET INTERRUPT ENABLE BIT IN XCSR
```

TEST - RESET AND INTERRUPT ENABLE BITS

```
19964 121420  012702  000340            MOV     #340,R2          ;SET PRIORITY TO 7
19965 121424  000402                    BR      7$               ;GO WAIT IN CASE OF INTERRUPTS
19966 121426  162702  000040      6$:   SUB     #40,R2           ;LOWER PRIORITY LEVEL
19967 121432  106402              7$:   MTPS    R2               ;SET PRIORITY
19968 121434  012703  000026            MOV     #26,R3           ;TIME DELAY
19969 121440  077301              8$:   SOB     R3,8$            ;WAIT FOR INTERRUPTS
19970 121442  000403                    BR      10$              ;IF INTERRUPTS DIDN'T HAPPENED, BRANCH
19971 121444  104101              9$:   ERROR   +101             ;INTERRUPTS HAPPEN AT WRONG PRIORITY
19972 121446  005726                    TST     (SP)+            ;CLEAN UP THE STACK
19973 121450  005726                    TST     (SP)+
19974 121452  022702  000200      10$:  CMP     #200,R2          ;AT PRIORITY 4?
19975 121456  001363                    BNE     6$               ;IF NOT LAST ONE, CONTINUE
19976 121460  106427  000340            MTPS    #340             ;RESTORE PRIORITY 7
19977 121464  042737  000100  177564    BIC     #BIT06,XCSR      ;CLEAR INTERRUPT ENABLE BIT
19978                                   ;
19979                                   ; CHECK THAT RECEIVE INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN 3
19980                                   ;
19981 121472  012737  121562  000060    MOV     #15$,@#60        ;POINT RECEIVE VECTOR TO PROGRAM AREA
19982 121500  012737  000340  000062    MOV     #340,@#62        ;AT PRIORITY 7
19983 121506  105737  177564      11$:  TSTB    XCSR             ;TRANSMITTER READY
19984 121512  100375                    BPL     11$              ;IF NOT, WAIT
19985 121514  012737  000000  177566    MOV     #NULL,XBUF       ;TRY TO TRANSMIT NULL
19986 121522  052737  000100  177560    BIS     #BIT06,RCSR      ;SET INTERRUPT ENABLE BIT IN RCSR
19987 121530  012702  000340            MOV     #340,R2          ;SET PRIORITY TO 7
19988 121534  000402                    BR      13$              ;GO WAIT IN CASE OF INTERRUPTS
19989 121536  162702  000040      12$:  SUB     #40,R2           ;LOWER PRIORITY LEVEL
19990 121542  052737  000004  177564  13$:  BIS     #BIT02,XCSR      ;SET LOOP BACK MODE
19991 121550  106402                    MTPS    R2               ;SET PRIORITY
19992 121552  012703  000100            MOV     #100,R3          ;TIME DELAY
19993 121556  077301              14$:  SOB     R3,14$           ;WAIT FOR INTERRUPTS
19994 121560  000406                    BR      16$              ;IF INTERRUPTS DIDN'T HAPPENED, BRANCH
19995 121562  042737  000004  177564  15$:  BIC     #BIT02,XCSR      ;CLEAR LOOP BACK MODE
19996 121570  104101                    ERROR   +101             ;INTERRUPTS HAPPEN AT WRONG PRIORITY
19997 121572  005726                    TST     (SP)+            ;CLEAN UP THE STACK
19998 121574  005726                    TST     (SP)+
19999 121576  022702  000200      16$:  CMP     #200,R2          ;AT PRIORITY 4?
20000 121602  001355                    BNE     12$              ;IF NOT LAST ONE, CONTINUE
20001                                   ;
20002                                   ; CLEAN UP BEFORE NEXT TEST
20003                                   ;
20004 121604  106427  000340            MTPS    #340             ;RESTORE PRIORITY 7
20005 121610  042737  000100  177560    BIC     #BIT06,RCSR      ;CLEAR INTERRUPT ENABLE BIT
20006 121616  012702  000300            MOV     #300,R2          ;STALL DELAY
20007 121622  105737  177560      17$:  TSTB    RCSR             ;RECEIVE READY?
20008 121626  100401                    BMI     18$              ;STOP WAITING, IF SO
20009 121630  077204                    SOB     R2,17$           ;OTHERWISE, STAY IN THE LOOP
20010 121632  005737  177562      18$:  TST     RBUF             ;READ CHARACTER TRANSMITTED
20011 121636  042737  000004  177564    BIC     #BIT02,XCSR      ;CLEAR LOOP BACK MODE
20012
```

```
                                .SBTTL  TEST  - INTERRUPT PRIORITY FOR SLU
                                ;CHECK THAT INTERRUPTS HAPPEN AT PRIORITY 3 AND THAT THEY CLEAR
                                ;RCSR<06> AND XCSR<06>.
                                ;
                                ;ROUTINE TEST
                                ;.      LET 60=#ADDRESS_OF_LEGAL_RINTERRUPT
                                ;.      LET 64=#ADDRESS_OF_LEGAL_XINTERRUPT
                                ;.      LET XCSR<02>=#1
                                ;.      SET PRIORITY TO #3
                                ;.      WAIT FOR XINTERRUPT=#3
                                ;.      IF XCSR<07> EQ #1 THEN
                                ;.          .       ERROR
                                ;.      ENDIF
                                ;.      WAIT FOR RINTERRUPT=#3
                                ;.      IF RCSR<07> EQ #0 THEN
                                ;.          .       ERROR
                                ;.      ENDIF
                                ;.      LET XCSR<02>=#0
                                ;.      SET PRIORITY TO NORMAL
                                ;ENDROUTINE
                                ;
                                ;ROUTINE LEGAL_XINTERRUPT
                                ;.      LET XBUF=#CHARACTER
                                ;.      INCREMENT XINTERRUPT
                                ;ENDROUTINE
                                ;
                                ;ROUTINE LEGAL_RINTERRUPT
                                ;.      READ RCSR
                                ;.      INCREMENT RINTERRUPT
                                ;ENDROUTINE

                                ;;*****************************************************************
    121644  000004              TST51:  SCOPE
20046 121646  122737  000001  001220          CMPB    #APTENV,$ENV                    ;RUNNING IN APT MODE?
20047 121654  001003                          BNE     100$                            ;NO, GO DO TEST
20048 121656  005737  001206                  TST     $PASS                           ;FIRST PASS?
20049 121662  001113                          BNE     TST52           ;;IF APT AND NOT FIRST PASS, EXIT TEST
20050                           ;
20051                           ; GET READY FOR INTERRUPTS
20052                           ;
20053 121664  012737  122060  000060  100$:   MOV     #8$,@#60                        ;STORE RECEIVER VECTOR
20054 121672  012737  122004  000064          MOV     #6$,@#64                        ;STORE TRANSMITTER VECTOR
20055 121700  012737  000340  000062          MOV     #340,@#62                       ;AT PRIORITY 7
20056 121706  012737  000340  000066          MOV     #340,@#66                       ;FOR RECEIVER AND TRANSMITTER
20057 121714  052737  000004  177564          BIS     #BIT02,XCSR                     ;SET LOOP BACK MODE
20058 121722  012701  000100                  MOV     #100,R1                         ;DELAY FOR UNEXPECTED CHARACTERS
20059 121726  105737  177560          1$:     TSTB    RCSR                            ;RECEIVER READY?
20060 121732  100401                          BMI     2$                              ;IF YES, BRANCH
20061 121734  077104                          SOB     R1,1$                           ;OTHERWISE, WAIT JUST IN CASE
20062 121736  005737  177562          2$:     TST     RBUF                            ;READ RECEIVER
20063                           ;
20064                           ; SET PRIORITIES AND XMIT INTERRUPTS
20065                           ;
20066 121742  012702  000140                  MOV     #140,R2                         ;START WITH PRIORITY 3
20067 121746  000402                          BR      4$                              ;TRY TO DO IT
20068 121750  162702  000040          3$:     SUB     #40,R2                          ;LOWER PRIORITY
20069 121754  106402                  4$:     MTPS    R2                              ;TRY TO DO AT LOWER PRIORITY
```

TEST - INTERRUPT PRIORITY FOR SLU

```
20070 121756  052737  000100  177564          BIS     #BIT06,XCSR                     ;LOOP BACK & INTERRUPT ENABLE
20071 121764  012703  001000                  MOV     #1000,R3                        ;WAIT DELAY FOR INTERRUPTS
20072 121770  077301                  5$:     SOB     R3,5$                           ;WAIT FOR XMIT INTERRUPTS
20073 121772  042737  000004  177564          BIC     #BIT02,XCSR                     ;CLEAR LOOP BACK BIT
20074 122000  104107                          ERROR   +107                            ;NO XMIT INTERRUPTS
20075 122002  000443                          BR      TST52           ;;IF ERROR, EXIT TEST
20076                                 ;
20077                                 ; TRANSMITTER INTERRUPT HERE
20078                                 ;
20079 122004  005726                  6$:     TST     (SP)+                           ;CLEAN UP STACK
20080 122006  005726                          TST     (SP)+
20081 122010  042737  000100  177564          BIC     #BIT06,XCSR                     ;CLEAR INTERRUPT ENABLE
20082 122016  012737  000000  177566          MOV     #NULL,XBUF                      ;TRANSMIT NULL
20083 122024  052737  000100  177560          BIS     #BIT06,RCSR                     ;SET RECEIVE INTERRUPT
20084 122032  106402                          MTPS    R2                              ;SET NEXT PRIORITY
20085 122034  012703  100000                  MOV     #100000,R3                      ;WAIT DELAY FOR INTERRUPTS
20086 122040  077301                  7$:     SOB     R3,7$                           ;WAIT FOR RECEIVE INTERUPTS
20087 122042  042737  000004  177564          BIC     #BIT02,XCSR                     ;CLEAR LOOP BACK MODE BIT
20088 122050  104107                          ERROR   +107                            ;NO RECEIVE INTERRUPTS
20089 122052  000406                          BR      9$                              ;DON'T TOUCH STACK
20090 122054  106427  000340                  MTPS    #340                            ;RAISE PRIORITY
20091                                 ;
20092                                 ; RECEIVER INTERRUPT HERE
20093                                 ;
20094 122060  005726                  8$:     TST     (SP)+                           ;CLEAN UP STACK
20095 122062  005726                          TST     (SP)+
20096 122064  005737  177562                  TST     RBUF                            ;READ RECEIVER BUFFER
20097 122070  005702                  9$:     TST     R2                              ;PRIORITY 0
20098 122072  001326                          BNE     3$                              ;IF NOT YET, CONTINUE
20099 122074  106427  000340                  MTPS    #340                            ;RAISE PRIORITY TO 7
20100 122100  042737  000100  177560          BIC     #BIT06,RCSR                     ;CLEAR RECEIVE INTER. ENABLE
20101 122106  005037  177564                  CLR     XCSR                            ;CLEAR XCSR
20102
```

```
20104                                   .SBTTL  TEST  - BREAK CONDITION
20105                                   ;CHECK THAT SENDING BREAK CAUSES FRAMING ERROR.
20106                                   ;
20107                                   ;RCSR   <15>    ERROR
20108                                   ;       <13>    FRAMING ERROR
20109                                   ;       <11>    RECEIVED BREAK
20110                                   ;
20111                                   ;XCSR   <00>    TRANSMIT BREAK
20112                                   ;
20113                                   ;ROUTINE TEST
20114                                   ;.      LET XCSR<02>=#1
20115                                   ;.      LET XCSR<00>=#1
20116                                   ;.      WAIT FOR RCSR<07>=#1
20117                                   ;.      IF RBUF<15!13!11> NE #1 THEN
20118                                   ;.      .       ERROR (ERROR, FRAMING ERROR, RECEIVE BREAK NE 1)
20119                                   ;.      ENDIF
20120                                   ;.      LET XCSR<00>=#0
20121                                   ;.      IF XCSR<00> NE #0 THEN
20122                                   ;.      .       ERROR (XCSR<00> DOES NOT GO LOW)
20123                                   ;.      ENDIF
20124                                   ;.      WAIT FOR XCSR<07>=#1
20125                                   ;.      LET XBUF=#NULL (SEND NULL CHARACTER TO SEE ERROR CLEARED)
20126                                   ;.      WAIT FOR RCSR<07>=#1
20127                                   ;.      IF RBUF<15!13!11> NE #0 THEN
20128                                   ;.      .       ERROR
20129                                   ;.      ENDIF
20130                                   ;.      LET XCSR<00>=#1
20131                                   ;.      EXECUTE "RESET"
20132                                   ;.      IF XCSR<00> NE #0 THEN
20133                                   ;.      .       ERROR
20134                                   ;.      ENDIF
20135                                   ;.      LET XCSR<02>=#0
20136                                   ;ENDROUTINE
20137
20138                                   ;;*****************************************************************
    122112  000004                      TST52:  SCOPE
20139                                   ;
20140                                   ; DECIDE WHETHER TO RUN THIS TEST
20141                                   ;
20142 122114  032737  000200  000052            BIT     #BIT07,@#52                     ;UFD MODE?
20143 122122  001127                            BNE     TST53           ;;IN UFD MODE, EXIT TEST
20144 122124  005737  001206                    TST     $PASS                           ;FIRST PASS?
20145 122130  001124                            BNE     TST53           ;;IF APT AND NOT FIRST PASS, EXIT TEST
20146                                   ;
20147                                   ; SEND BREAK AND CHECK ERROR BITS IN RBUF
20148                                   ;
20149 122132  052737  000004  177564    1$:     BIS     #BIT02,XCSR                     ;TRANSMIT IN LOOP BACK
20150 122140  013737  177520  002730            MOV     BCSR,SAVBR                      ;SAVE BCSR
20151 122146  042737  001000  177520            BIC     #BIT09,BCSR                     ;DISABLE HALT ON BREAK
20152 122154  052737  000001  177564            BIS     #BIT00,XCSR                     ;SET SEND BREAK BIT
20153 122162  032737  000001  177564            BIT     #BIT00,XCSR                     ;GOT SET OK?
20154 122170  001001                            BNE     2$                              ;IF YES, BRANCH
20155 122172  104110                            ERROR   +110                            ;WRITING 1 TO XCSR<0>
20156 122174  012701  000100            2$:     MOV     #100,R1                         ;STALL DELAY
20157 122200  105737  177560            4$:     TSTB    RCSR                            ;RECEIVER READY?
20158 122204  100401                            BMI     5$                              ;IF YES, BRANCH
20159 122206  077104                            SOB     R1,4$                           ;WAIT JUST IN CASE OF A CHARACTER
```

TEST - BREAK CONDITION

```
20160 122210  005737  177562          5$:     TST     RBUF                            ;READ A CHARACTER
20161 122214  052737  000001  177564          BIS     #BIT00,XCSR                     ;TRANSMIT BREAK
20162 122222  012701  001000                  MOV     #1000,R1                        ;ANOTHER DELAY TO GET BREAK
20163 122226  077101                  6$:     SOB     R1,6$                           ;WAIT A WHILE
20164 122230  105737  177560          7$:     TSTB    RCSR                            ;RECEIVER READY?
20165 122234  100375                          BPL     7$                              ;IF NOT, WAIT
20166 122236  013737  177562  001126          MOV     RBUF,$BDDAT                     ;STORE WHATEVER RECEIVED
20167 122244  022737  124000  001126          CMP     #BIT15!BIT13!BIT11,$BDDAT       ;ALL ERROR BITS SET?
20168 122252  001405                          BEQ     8$                              ;IF YES, BRANCH
20169 122254  042737  000004  177564          BIC     #BIT02,XCSR                     ;RESET TO ENABLE SLU
20170 122262  104105                          ERROR   +105                            ;BREAK DOES NOT CAUSE ERRORS
20171 122264  000446                          BR      TST53           ;;EXIT
20172 122266  042737  000001  177564  8$:     BIC     #BIT00,XCSR                     ;CLEAR TRANSMIT BREAK
20173 122274  032737  000001  177564          BIT     #BIT00,XCSR                     ;GOT CLEARED OK?
20174 122302  001405                          BEQ     9$                              ;IF YES, BRANCH
20175 122304  042737  000004  177564          BIC     #BIT02,XCSR                     ;RESET TO ENABLE SLU
20176 122312  104110                          ERROR   +110                            ;ERROR WRITING 0 TO XCSR<0>
20177 122314  000432                          BR      TST53           ;;EXIT
20178                                 ;
20179                                 ; CHECK THAT BREAK CONDITION IS CLEARED
20180                                 ;
20181 122316  013737  002730  177520  9$:     MOV     SAVBR,BCSR                      ;RESTORE BCSR
20182 122324  105737  177564          10$:    TSTB    XCSR                            ;XMIT READY?
20183 122330  100375                          BPL     10$                             ;IF NOT, WAIT
20184 122332  012737  000177  177566          MOV     #177,XBUF                       ;TRY TO TRANSMIT DELETE
20185 122340  105737  177560          11$:    TSTB    RCSR                            ;RECEIVER READY
20186 122344  100375                          BPL     11$                             ;IF NOT, WAIT
20187 122346  013737  177562  001126          MOV     RBUF,$BDDAT                     ;STORE RECEIVE BUFFER
20188 122354  032737  124000  001126          BIT     #BIT15!BIT13!BIT11,$BDDAT       ;ERRORS CLEARED?
20189 122362  001404                          BEQ     12$                             ;IF YES, BRANCH
20190 122364  042737  000004  177564          BIC     #BIT02,XCSR                     ;RESET TO ENABLE SLU
20191 122372  104106                          ERROR   +106                            ;BREAK NOT CLEARED ON NEXT CHARACTER
20192 122374  042737  000004  177564  12$:    BIC     #BIT02,XCSR                     ;CLEAR LOOP BACK MODE
20193
20194
```

```
                                .SBTTL  TEST  - OVERRUN CONDITION
                                ;CHECK OVERRUN CONDITION
                                ;
                                ;RCSR   <14>    OVERRUN ERROR
                                ;
                                ;ROUTINE TEST
                                ;.      LET XCSR<02>=#1 (LOOPBACK MODE)
                                ;.      WAIT FOR XCSR<07>=#1
                                ;.      LET XBUF=#252
                                ;.      WAIT FOR XCSR<07>=#1
                                ;.      LET XBUF=#125 (SEND THE 2ND W/O READING THE 1ST CHARACTER)
                                ;.      WAIT FOR RCSR<07>=#1
                                ;.      STALL FOR LOWEST BAUD RATE TO GET 2ND CHARACTER
                                ;.      IF LOW BYTE OF RBUF NE #125 THEN
                                ;.      .       ERROR (1ST CHARACTER WASN'T OVERRUN)
                                ;.      ENDIF
                                ;.      IF RBUF<15!14> NE #1 THEN
                                ;.      .       ERROR (NO OVERRUN BIT SET)
                                ;.      ENDIF
                                ;.      WAIT FOR XCSR<07>=#1
                                ;.      LET XBUF=#NULL
                                ;.      WAIT FOR RCSR<07>=#1
                                ;.      IF RBUF<15!14> NE #0 THEN
                                ;.      .       ERROR (WASN'T CLEARED ON THE NEXT CHARACTER RECEIVED)
                                ;.      ENDIF
                                ;.      LET XCSR<02>=#0
                                ;ENDROUTINE

                                ;;*****************************************************************
     122402  000004             TST53:  SCOPE
20225 122404  122737  000001  001220        CMPB    #APTENV,$ENV                    ;RUNNING IN APT MODE?
20226 122412  001003                        BNE     100$                            ;NO, GO DO TEST
20227 122414  005737  001206                TST     $PASS                           ;FIRST PASS?
20228 122420  001077                        BNE     TST54           ;;IF APT AND NOT FIRST PASS, EXIT TEST
20229 122422  052737  000004  177564  100$:  BIS     #BIT02,XCSR                     ;SET LOOP BACK MODE
20230 122430  105737  177564        1$:     TSTB    XCSR                            ;READY TO TRANSMIT?
20231 122434  100375                        BPL     1$                              ;IF NOT, WAIT
20232 122436  012737  000021  177566        MOV     #21,XBUF                        ;TRANSMIT A CHARACTER
20233 122444  105737  177560        2$:     TSTB    RCSR                            ;RECEIVE READY?
20234 122450  100375                        BPL     2$                              ;IF NOT, WAIT
20235 122452  105737  177564        3$:     TSTB    XCSR                            ;READY TO TRANSMIT?
20236 122456  100375                        BPL     3$                              ;IF NOT, WAIT
20237 122460  012737  000177  177566        MOV     #177,XBUF                       ;TRANSMIT THE 2ND CHARACTER
20238 122466  012703  150000                MOV     #150000,R3                      ;STALL FOR THE 2ND CHARACTER
20239 122472  077301                4$:     SOB     R3,4$                           ;WAIT A WHILE
20240 122474  013737  177562  001126        MOV     RBUF,$BDDAT                     ;STORE RECEIVED DATA
20241 122502  012737  140177  001124        MOV     #140177,$GDDAT                  ;EXPETED PATTERN
20242 122510  122737  000177  001126        CMPB    #177,$BDDAT                     ;2ND CHARACTER RECEIVED?
20243 122516  001404                        BEQ     5$                              ;IF YES, BRANCH
20244 122520  042737  000004  177564        BIC     #BIT02,XCSR                     ;RESET TO ENABLE SLU
20245 122526  104111                        ERROR   +111                            ;2ND CHARACTER DIDN'T OVERRUN 1ST
20246 122530  122737  000300  001127  5$:   CMPB    #BIT7!BIT6,$BDDAT+1             ;OVERRUN ERROR BITS SET?
20247 122536  001404                        BEQ     6$                              ;IF YES, BRANCH
20248 122540  005037  177564                CLR     XCSR                            ;RESET TO ENABLE SLU
20249 122544  104112                        ERROR   +112                            ;OVERRUN DOES NOT SET ERRORS BITS
20250 122546  000424                        BR      TST54           ;;EXIT
20251                           ;
```

TEST - OVERRUN CONDITION

```
20252                                   ; SEND NEXT CHARACTER TO CLEAR OVERRUN CONDITIONS
20253                                   ;
20254 122550  105737  177564      6$:     TSTB    XCSR                    ;TRANSMITTER READY?
20255 122554  100375                      BPL     6$                      ;IF NOT, BRANCH AND WAIT
20256 122556  012737  000000  177566      MOV     #NULL,XBUF              ;TRANSMIT NULL CHARACTER
20257 122564  105737  177560      7$:     TSTB    RCSR                    ;RECEIVER READY?
20258 122570  100375                      BPL     7$                      ;IF NOT, BRANCH AND WAIT
20259 122572  032737  140000  177562      BIT     #BIT15!BIT14,RBUF       ;ANY ERRORS SET?
20260 122600  001404                      BEQ     8$                      ;IF NOT, BRANCH
20261 122602  042737  000004  177564      BIC     #BIT02,XCSR             ;RESET TO ENABLE SLU
20262 122610  104113                      ERROR   +113                    ;OVERRUN NOT CLEARED ON NEXT CHAR.
20263 122612  042737  000004  177564  8$: BIC     #BIT02,XCSR             ;CLEAR LOOP BACK MODE BIT
20264
20265 122620                          SLEND:                            ;LAST SLU TEST
```

TEST - LED'S ON

```
20267                                   .SBTTL  TEST  - LED'S ON
20268                                   ;LED'S ON
20269                                   ;THIS TEST WILL INITIALIZE BDR TO CONTAIN A ROTATING PATTERN
20270                                   ;DISPLAYED IN LED'S.
20271                                   ;
20272                                   ;ROUTINE TEST
20273                                   ;.      WHILE A KEY NOT RECEIVED FROM KEYBOARD DO
20274                                   ;.      .       STALL ALLOWING TIME TO SEE PATTERN
20275                                   ;.      .       ROTATE LEFT TO LIGHT UP NEXT LED'S
20276                                   ;.      ENDDO
20277                                   ;ENDROUTINE
20278
20279                                   ;;*****************************************************************
     122620  000004                   TST54:  SCOPE
20280 122622  052737  001000  177520           BIS     #1000,BCSR              ;ENABLE HOB FOR APT
20281 122630  005005                           CLR     R5                      ;FLAG IN NO INTERRRUPT MODE
20282 122632  032737  000001  000052           BIT     #BIT00,@#52             ;IF RUNNING IN CHAIN MODE
20283 122640  001427                           BEQ     1$                      ;SKIP PRINTOUTS
20284 122642  005737  001206                   TST     $PASS                   ;1ST PASS?
20285 122646  001024                           BNE     1$                      ;IF NOT, SKIP PRINTOUTS
20286 122650  122737  000001  001220           CMPB    @APTENV,$ENV            ;APT MODE?
20287 122656  001420                           BEQ     1$                      ;YES, SKIP PRINTOUT'S
20288 122660  005105                           COM     R5                      ;CLEAR FLAG IN INTERRUPT MODE
20289 122662  104401  001175                   TYPE    ,$CRLF
20290 122666  104401  123016                   TYPE    ,LEDS                   ;IDENTIFY THE TEST
20291 122672  012737  122770  000060           MOV     #5$,@#60                ;RECEIVE SLU VECTOR
20292 122700  012737  000340  000062           MOV     #340,@#62               ;AT PRIORITY 7
20293 122706  052737  000100  177560           BIS     #BIT06,RCSR             ;ENABLE INTERRUPTS
20294 122714  106427  000140                   MTPS    #140                    ;LOWER PRIORITY
20295 122720  012704  000006           1$:     MOV     #6,R4                   ;FOR EACH LOOP
20296 122724  012701  000076                   MOV     #76,R1                  ;START WITH 1
20297 122730  110137  177524           2$:     MOVB    R1,BDR                  ;TURN OFF FIRST LED
20298 122734  012703  000004                   MOV     #4,R3                   ;STALL DELAY
20299 122740  012702  177777           3$:     MOV     #177777,R2              ;STALL DELAY
20300 122744  077201                   4$:     SOB     R2,4$                   ;WAIT A WHILE
20301 122746  077304                           SOB     R3,3$                   ;WAIT A WHILE
20302 122750  000261                           SEC                             ;SET CARRY
20303 122752  006101                           ROL     R1                      ;GET ANOTHER LED
20304 122754  077413                           SOB     R4,2$                   ;DO A FEW TIMES
20305 122756  005705                           TST     R5                      ;RUNNING IN INTERACTIVE MODE?
20306 122760  001407                           BEQ     6$                      ;IF NOT, EXIT
20307 122762  104401  001170                   TYPE    ,$BELL                  ;
20308 122766  000754                           BR      1$                      ;REPEAT PATTERN
20309 122770  005737  177562           5$:     TST     RBUF                    ;READ BUFFER
20310 122774  062706  000004                   ADD     #4,SP                   ;ADJUST STACK
20311 123000  112737  000377  177524   6$:     MOVB    #377,BDR                ;NO MORE
20312 123006  042737  001000  177520           BIC     #1000,BCSR              ;DISABLE HOB FOR APT
20313 123014  000452                           BR      TST55           ;;EXIT TEST
20314
20315 123016     012     015     124   LEDS:   .ASCII  <12><15>/THIS IS A TEST FOR ON-BOARD LED'S/<12><15>
20316 123063     124     131     120           .ASCIZ  /TYPE ANY CHARACTER ON A KEYBOARD TO CONTINUE/<12><15>
20317                                   .EVEN
20318
```

```
                                .SBTTL  TEST  - MEMORY MAPPING
                                ;MEMORY MAPPING
                                ;THIS TEST WILL AUTOSIZE MEMORY IN 2K BYTES. EVERY PAGE WILL BE
                                ;CHECKED FOR WHAT TYPE IT IS: Q-BUS, UNIBUS OR PMI BUS MEMORY BY
                                ;SEEING HOW IT CAN BE CACHED.
                                ;ROUTINE TEST
                                ;.  LET 4=ADDRESS OF NON-EXISTENT_MEMORY
                                ;.  IF KMCR<05-00> NE #1 THEN (NOT ALL UNIBUS MEMORY)
                                ;.      TURN ON MMU AND REMAP THE PROGRAM AREA
                                ;.      REPEAT
                                ;.      .  DO FOR KDPAR0 FROM #0 TO #177600
                                ;.      .       DO FOR R1 FROM #0 TO #20000 BY #4000
                                ;.      .       .       READ (R1)
                                ;.      .       .       IF ABORT_NON-EXISTENT_MEMORY NE 1
                                ;.      .       .       .       MEMORY EXISTS
                                ;.      .       .       .       READ (R1)+
                                ;.      .       .       .       READ (R1)
                                ;.      .       .       .       IF HIT/MISS EQ 2_HITS THEN
                                ;.      .       .       .       .       PMI MEMORY
                                ;.      .       .       .       ELSE
                                ;.      .       .       .       .       IF HIT/MISS EQ HIT THEN
                                ;.      .       .       .       .       .       Q-BUS MEMORY
                                ;.      .       .       .       .       ELSE
                                ;.      .       .       .       .       .       ERROR
                                ;.      .       .       .       .       ENDIF
                                ;.      .       .       .       ENDIF
                                ;.      .       .       LET ABORT_NON-EXISTENT_MEMORY=#0
                                ;.      .       ENDDO
                                ;.      .  ENDDO
                                ;.      UNTILL ABORT_NON-EXISTANT_MEMORY EQ #1
                                ;.  ENDIF
                                ;.
                                ;ENDROUTINE
                                ;
                                ;ROUTINE NON-EXISTENT_MEMORY
                                ;.      LET ABORT_NON-EXISTENT_MEMORY=#1
                                ;.      RETURN
                                ;ENDROUTINE

                                ;;*****************************************************************
    123142  000004              TST55:  SCOPE
20361 123144  032737  000001  000052        BIT     #BIT00,@#52                     ;CHAIN MODE?
20362 123152  001561                        BEQ     TST56           ;;IF SO, EXIT TEST
20363 123154  005737  001206                TST     $PASS                           ;FIRST PASS?
20364 123160  001156                        BNE     TST56           ;;IF APT AND NOT FIRST PASS, EXIT TEST
20365 123162  122737  000001  001220        CMPB    #APTENV,$ENV                    ;APT MODE?
20366 123170  001552                        BEQ     TST56           ;;YES, SKIP PRINTOUT'S
20367                           ;
20368                           ; SETUP ALL REGISTERS FOR MAPPING
20369                           ;
20370 123172  012737  000400  177746        MOV     #400,CCR                        ;FLUSH THE CACHE
20371 123200  013704  000004                MOV     ERRVEC,R4                       ;STORE TIMEOUT VECTOR
20372 123204  012737  123354  000004        MOV     #7$,@#ERRVEC                    ;POINT TO PROGRAM AREA
20373 123212  012737  000340  000006        MOV     #340,ERRVEC+2                   ;AT PRIORITY 4
20374 123220  004737  136574                JSR     PC,INITMM                       ;REMAP PROGRAM AREA
20375 123224  005037  172354                CLR     KIPAR6                          ;USED FOR MAPPING MEMORY
```

```
20376 123230  005002                        CLR     R2              ;CLEAR PAGE COUNT FOR PMI
20377 123232  005003                        CLR     R3              ;CLEAR PAGE COUNT FOR QBUS
20378 123234  005237  177572                INC     MMR0            ;ENABLE MEMORY MANGEMENT
20379 123240  052737  000020  172516        BIS     #BIT04,MMR3     ;ENABLE 22 BITS
20380 123246  000403                        BR      2$              ;GO ACCESS
20381                               ;
20382                               ; TRY TO MAP ALL PAGES
20383                               ;
20384 123250  062737  000200  172354  1$:   ADD     #200,KIPAR6     ;INCREMENT BY 4K WORDS
20385 123256  012701  140000          2$:   MOV     #140000,R1      ;ACCESS THRU KIPAR6
20386 123262  000402                        BR      4$              ;START DOING IT
20387 123264  062701  003776          3$:   ADD     #3776,R1        ;INCREMENT BY 1K WORDS
20388 123270  005721                  4$:   TST     (R1)+           ;ACCESS 1ST LOCATION
20389 123272  042737  001000  177520        BIC     #1000,BCSR      ;DISABLE HALT ON BREAK
20390 123300  005711                        TST     (R1)            ;ACCESS 2ND LOCATION
20391 123302  013737  177752  114150        MOV     HITMIS,RECDAT   ;STORE REGISTER
20392 123310  052737  001000  177520        BIS     #1000,BCSR      ;ENABLE HALT ON BREAK
20393 123316  032737  000004  114150        BIT     #BIT02,RECDAT   ;LAST (R1) HIT?
20394 123324  001402                        BEQ     5$              ;IF NOT, QBUS MEMORY
20395 123326  005202                        INC     R2              ;INCREMENT 1K COUNT FOR PMI
20396 123330  000401                        BR      6$              ;GO CONITNUE
20397 123332  005203                  5$:   INC     R3              ;INCREMENT COUNT FOR QBUS MEM.
20398 123334  022701  154000          6$:   CMP     #154000,R1      ;LAST IN 4K PAGE?
20399 123340  101351                        BHI     3$              ;IF NOT, BRANCH
20400 123342  022737  177600  172354        CMP     #177600,KIPAR6  ;2M BOUNDARY?
20401 123350  001337                        BNE     1$              ;IF NOT, BRANCH
20402 123352  000402                        BR      8$              ;IF 2M, DON'T TOUCH STACK
20403                               ;
20404                               ; MAPPING IS DONE, FIND OUT HOW MANY PAGES WERE THERE
20405                               ;
20406 123354  005726                  7$:   TST     (SP)+           ;ADJUST STACK
20407 123356  005726                        TST     (SP)+
20408 123360  010437  000004          8$:   MOV     R4,ERRVEC       ;RESTORE ERROR VECTOR
20409 123364  005037  177572                CLR     MMR0            ;DISABLE MEMORY MANGEMENT
20410 123370  042737  000020  172516        BIC     #BIT04,MMR3     ;DISABLE 22 BITS
20411 123376  005702                        TST     R2              ;ANY PMI MEMORY?
20412 123400  001405                        BEQ     9$              ;IF NOT, GO CHECK QBUS MEMORY
20413 123402  006302                        ASL     R2              ;TRANSLATE TO BYTES
20414 123404  010246                        MOV     R2,-(SP)        ;STORE 1K # ON STACK
20415 123406  104405                        TYPDS                   ;TYPE # OF PAGES
20416 123410  104401  123434                TYPE    ,MEMK           ;TYPE ASCII
20417 123414  005703                  9$:   TST     R3              ;ANY Q-BUS MEMORY?
20418 123416  001405                        BEQ     10$             ;IF NOT, BRANCH
20419 123420  006303                        ASL     R3              ;TRANSLATE TO BYTES
20420 123422  010346                        MOV     R3,-(SP)        ;STORE 1K # ON STACK
20421 123424  104405                        TYPDS                   ;TYPE # OF PAGES
20422 123426  104401  123464                TYPE    ,MEMQ           ;TYPE ASCII
20423 123432                          10$:
      123432  000431                        BR      TST56           ;;EXIT TEST
20424
20425 123434     113     040     102  MEMK:  .ASCIZ  /K BYTES OF PMI MEMORY/<12><15>
20426 123464     113     040     102  MEMQ:  .ASCIZ  /K BYTES OF Q-BUS MEMORY/<12><15>
20427                                 .EVEN
20428
20429                                 .SBTTL WRONG PARITY ABORT TEST
20430                                 ;WRONG PARITY ABORT
20431                                 ;THIS TEST VERIFIES ABORT TO 114 USING PARITY OR ECC MEMORY CSR.
```

WRONG PARITY ABORT TEST

```
20432                                   ;IF MORE THEN 1 CSR PRESENT, ALL OF THEM WILL BE WRITTEN AT THE
20433                                   ;SAME TIME.
20434                                   ;
20435                                   ;ROUTINE TEST
20436                                   ;.      SIZE FOR ALL POSSIBLE MEMORY CSR'S
20437                                   ;.      STORE UP TO 16 CSR STARTING FROM TEMP
20438                                   ;.      WRITE ALL WITH WRONG PARITY
20439                                   ;.      WRITE TO 0
20440                                   ;.      READ IT BACK
20441                                   ;.      IF NO ABORT TO 114 THEN
20442                                   ;.      .       ERROR IN PARITY ABORT LOGIC
20443                                   ;.      ENDIF
20444                                   ;.      RESTORE CSR'S
20445                                   ;ENDROUTINE
20446
20447                                   ;;*****************************************************************
     123516  000004                     TST56:  SCOPE
20448                                   ;
20449                                   ; FIND OUT ALL POSSIBLE MEMORY CSR LOCATIONS UP TO 16
20450                                   ;
20451 123520  013701  000004                      MOV     ERRVEC,R1               ;STORE TIMEOUT VECTOR
20452 123524  012737  123562  000004            MOV     #2$,ERRVEC              ;POINT NEW TO PROGRAM
20453 123532  012737  000340  000006            MOV     #340,ERRVEC+2           ;AT PRIORITY 7
20454 123540  005004                            CLR     R4                      ;COUNT FOR CSR'S
20455 123542  012702  172100                    MOV     #172100,R2              ;FIRST POSSIBLE CSR
20456 123546  012703  002740                    MOV     #TEMP,R3                ;STORAGE LOCATION
20457 123552  005712                  1$:       TST     (R2)                    ;IS CSR THERE?
20458 123554  010223                            MOV     R2,(R3)+                ;IF THERE, STORE
20459 123556  005204                            INC     R4                      ;INCREMENT COUNT FOR CSR'S
20460 123560  000402                            BR      3$                      ;BRANCH AROUND
20461 123562  005726                  2$:       TST     (SP)+                   ;RESTORE STACK
20462 123564  005726                            TST     (SP)+
20463 123566  062702  000002          3$:       ADD     #2,R2                   ;POINT TO NEW CSR
20464 123572  022702  172136                    CMP     #172136,R2              ;ALL DONE?
20465 123576  101365                            BHI     1$                      ;IF NOT, BRANCH
20466 123600  010137  000004                    MOV     R1,ERRVEC               ;RESTORE ERROR VECTOR
20467 123604  052737  001000  177746            BIS     #BIT09,CCR              ;SET CACHE BYPASS
20468                                   ;
20469                                   ; WRITE ALL CSR'S WITH WRONG ECC CODE
20470                                   ; NOTE: IN PARITY MEMORY THOSE BITS ARE READ ONLY AND
20471                                   ;       DIAGNOSTIC MODE BIT FOR ECC IS THE SAME AS WRONG PARITY
20472                                   ;
20473 123612  013701  000114                    MOV     @#114,R1                ;STORE PARITY ABORT VECTOR
20474 123616  012737  123704  000114            MOV     #6$,@#114               ;POINT NEW TO PROGRAM
20475 123624  012737  000340  000116            MOV     #340,@#116              ;AT PRIORITY 7
20476 123632  010402                            MOV     R4,R2                   ;STORE CSR COUNT
20477 123634  012703  002740                    MOV     #TEMP,R3                ;START WITH 1ST CSR
20478 123640  052773  000005  000000  4$:       BIS     #BIT02!BIT00,@0(R3)     ;DIAGNOSTIC OR WRONG PARITY
20479 123646  042733  003740                    BIC     #3740,@(R3)+            ;CLEAR <10-5> CHECK BITS
20480 123652  077406                            SOB     R4,4$                   ;DO FOR ALL CSR'S
20481 123654  005037  000000                    CLR     @#0                     ;CLEAR TEST LOCATION WITH WRONG PR
20482 123660  010204                            MOV     R2,R4                   ;RESTORE COUNTER
20483 123662  012703  002740                    MOV     #TEMP,R3                ;START WITH 1ST CSR
20484 123666  042733  000004          5$:       BIC     #BIT02,@(R3)+           ;CLEAR ALL WRONG PARITY
20485 123672  077203                            SOB     R2,5$                   ;IN ALL CSR'S
20486 123674  005737  000000                    TST     @#0                     ;READ BACK WRONG PARITY
20487 123700  104034                            ERROR   +34                     ;NO WRONG PARITY ABORT
```

WRONG PARITY ABORT TEST

```
20488 123702  000402                        BR      7$
20489 123704  005726                6$:     TST     (SP)+                   ;ADJUST STACK
20490 123706  005726                        TST     (SP)+
20491 123710  012703  002740        7$:     MOV     #TEMP,R3                ;START WITH 1ST CSR
20492 123714  042733  000001        8$:     BIC     #BIT00,@(R3)+           ;CLEAR ALL WRONG PARITY
20493 123720  077403                        SOB     R4,8$                   ;IN ALL CSR'S
20494 123722  032737  100000  177744        BIT     #BIT15,MSER             ;ABORT REFLECTED IN MSER?
20495 123730  001004                        BNE     9$                      ;IF YES, BRANCH
20496 123732  012737  100000  001124        MOV     #100000,$GDDAT          ;
20497 123740  104035                        ERROR   +35                     ;MSER<15> NOT SET ON PARITY ABORT
20498 123742  042737  001000  177746  9$:   BIC     #BIT09,CCR              ;CLEAR CACHE BYPASS
20499 123750  005037  000000              CLR     @#0                     ;CLEAR WRONG PARITY
20500 123754  005037  177744              CLR     MSER                    ;CLEAR MSER
20501 123760  010137  000114              MOV     R1,@#114                ;RESTORE PARITY ABORT
20502
20503
```

TEST - DMA TAG PARITY IN STANDALONE MODE

```
                                .SBTTL  TEST  - DMA TAG PARITY IN STANDALONE MODE
                                ;CHECK DMA TAG STORE PARITY BIT.
                                ;ROUTINE TEST
                                ;.      CACHE DMA_PARITY
                                ;.      LET BCSR<08>=#1
                                ;.      REPORT ALL ERRORS
                                ;ENDROUTINE
                                ;
                                ;ROUTINE DMA_PARITY
                                ;.      GENERATE PARITY ERRORS
                                ;.      CHECK MSER<13>
                                ;.      LET BCSR<08>=#0
                                ;ENDROUTINE

                                ;;*****************************************************************
 123764  000004                 TST57:  SCOPE
123766  000240                          NOP
123770  005737  003032                 TST     CCHPAS                  ;have done enough inclusive passes?
123774  001002                          BNE     99$                     ; not yet
123776  000240                          NOP                             ; debug aid
124000  000456                          BR      TST60           ;;GO TO NEXT TEST
124002  000240                  99$:    NOP
                                ;
                                ; ALLOCATE CODE IN CACHE
                                ;
124004  012702  124040                  MOV     #DMAPAR,R2              ;POINT TO STANDALONE CODE
124010  042737  001000  177520          BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
124016  005722                  1$:     TST     (R2)+                   ;ALLOCATE IN CACHE
124020  022702  124100                  CMP     #DPAREN,R2              ;LAST ADDRESS?
124024  001374                          BNE     1$                      ;IF NOT, BRANCH
124026  005737  002740                  TST     TEMP                    ;ALLOCATE TEST LOCATION
124032  013737  177520  002730          MOV     BCSR,SAVBR              ;SAVE BCSR
                                ;
                                ; IN STANDALONE MODE TRY TO VERIFY RESPONSE TO PARITY ERRORS
                                ;
124040  052737  000400  177520  DMAPAR: BIS     #BIT08,BCSR             ;SET STANDALONE MODE BIT
124046  052737  002001  177746          BIS     #BIT10!BIT00,CCR        ;WRITE WRONG TAG PARITY
124054  005037  002740                  CLR     TEMP                    ;WRITE HIT WITH WRONG PARITY
124060  013705  177744                  MOV     MSER,R5                 ;STORE MSER
124064  042737  002000  177746          BIC     #BIT10,CCR              ;CLEAR WRONG PARITY BIT
124072  042737  000400  177520  2$:     BIC     #BIT08,BCSR             ;CLEAR STANDALONE BIT
                                ;
                                ; RETURN FROM STANDALONE MODE
                                ;
124100                          DPAREN:
124100  022705  060020                  CMP     #60020,R5               ;WRONG DMA PARITY?
124104  001401                          BEQ     4$                      ;IF OK, BRANCH
124106  104117                          ERROR   +117                    ;MSER NOT SET IN STANDALONE MODE
124110  005037  177744          4$:     CLR     MSER
124114  012737  000400  177746          MOV     #400,CCR                ;FLUSH THE CACHE
124122  013737  002730  177520          MOV     SAVBR,BCSR              ;RESTORE BCSR
124130  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
```

```
20558                                   .SBTTL  TEST  - DMA TAG PARITY W/O STANDALONE MODE
20559                                   ;CHECK DMA TAG PARITY BIT WITHOUT GOING INTO STANDALONE MODE.
20560                                   ;
20561                                   ;CCR    <10>    WRITE WRONG TAG PARITY
20562                                   ;
20563                                   ;ROUTINE TEST
20564                                   ;.      LET CCR<10>=#1
20565                                   ;.      ALLOCATE LOCATION IN CACHE
20566                                   ;.      INITIATE DMA WRITE TRANSFERS
20567                                   ;.      IF MSER<04> NE #1 THEN
20568                                   ;.          .       ERROR
20569                                   ;.      ENDIF
20570                                   ;ENDROUTINE
20571
20572                                   ;;*****************************************************************
     124136  000004                     TST60:  SCOPE
20573 124140  000240                             NOP
20574 124142  005737  003032                     TST     CCHPAS                  ;have done enough inclusive passes?
20575 124146  001002                             BNE     99$                     ; not yet
20576 124150  000240                             NOP                             ; debug aid
20577 124152  000471                             BR      TST61           ;;GO TO NEXT TEST
20578 124154  000240                     99$:    NOP
20579
20580 124156  032737  000200  000052             BIT     #BIT07,@#52                     ;UFD MODE?
20581 124164  001402                             BEQ     110$                            ;IF NOT, BRANCH
20582 124166  000137  140340                     JMP     $EOP                            ;OTHERWISE, NEXT PASS
20583 124172  005737  002664             110$:   TST     CSR1                            ;AT LEAST ONE Q22BE FOUND?
20584 124176  001457                             BEQ     TST61           ;;IF NOT, EXIT TEST
20585                                   ;
20586                                   ; ALLOCATE TEST IN CACHE
20587                                   ;
20588 124200  012703  124200             11$:    MOV     #.,R3                           ;START WITH CURRENT INSTRUCTION
20589 124204  005723                     10$:    TST     (R3)+                           ;READ A WORD
20590 124206  022703  124316                     CMP     #4$,R3                          ;ALL DONE?
20591 124212  001374                             BNE     10$                             ;IF NOT, CONTINUE
20592                                   ;
20593                                   ; WRITE A WORD WITH WRONG PARITY
20594                                   ;
20595 124214  013737  000114  001160     1$:     MOV     @#114,$TMP0                     ;STORE PARITY VECTOR
20596 124222  012737  124270  000114             MOV     #2$,@#114                       ;POINT TO TEST AREA
20597 124230  052737  002000  177746             BIS     #BIT10,CCR                      ;WRITE WRONG TAG PARITY
20598 124236  005037  002740                     CLR     TEMP                            ;WRITE MISS WITH WRONG TAG PARITY
20599 124242  042737  002000  177746             BIC     #BIT10,CCR                      ;CLEAR WRONG TAG PARITY
20600 124250  052737  000200  177746             BIS     #BIT07,CCR                      ;PARITY ABORT
20601 124256  005000                             CLR     R0                              ;FLAG TO DO 1 TRANSFER
20602 124260  004737  137370                     JSR     PC,DMATRN                       ;DO DMA WRITE TO TEMP THRU Q22BE
20603 124264  104116                             ERROR   +116                            ;NO PARITY ABORT
20604 124266  000413                             BR      4$                              ;BRANCH TO TEST MSER
20605 124270  013704  177744             2$:     MOV     MSER,R4                         ;STORE REGISTER
20606 124274  005037  177744                     CLR     MSER                            ;CLEAR MSER
20607 124300  005726                             TST     (SP)+                           ;
20608 124302  005726                             TST     (SP)+                           ;RESTORE STACK
20609 124304  005726                             TST     (SP)+
20610 124306  032704  000020             3$:     BIT     #BIT04,R4                       ;MSER OK?
20611 124312  001001                             BNE     4$                              ;IF SET, BRANCH
20612 124314  104116                             ERROR   +116                            ;MSER<4> NOT SET
20613 124316  005037  177744             4$:     CLR     MSER                            ;CLEAR MSER
```

TEST - DMA TAG PARITY W/O STANDALONE MODE

```
 20614 124322  012737  000400  177746          MOV     #400,CCR                        ;FLUSH CACHE
 20615 124330  013737  001160  000114          MOV     $TMP0,@#114                     ;RESTORE VECTOR
```

```
                                .SBTTL  TEST  - DMA WRITE HIT CYCLES
                                ;CHECK THAT DMA WRITE HITS INVALIDATE CACHE.
                                ;ROUTINE TEST
                                ;.      ALLOCATE A LOCATION IN CACHE
                                ;.      INITIATE DMA WRITE TO THIS LOCATION
                                ;.      READ THIS LOCATION BACK
                                ;.      IF IT IS CHANGED OR HIT/MISS EQ HIT
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;.      WRITE TO THE FIRST 16 LOCATION THEIR ADDRESS
                                ;.      DO BLOCK MODE TRANSFER TO THOSE LOCATIONS
                                ;.      START READ WITH THE LAST LOCATION
                                ;.      IF RECORD ANY HITS THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;.      INITIATE READ DMA TO THE SAME TEST LOCATIONS
                                ;.      READ THEM BACK
                                ;.      IF HIT/MISS NE HIT THEN
                                ;.      .       ERROR
                                ;.      ENDIF
                                ;ENDROUTINE

                                ;;*****************************************************************
 124336  000004                 TST61:  SCOPE
20640 124340  000240                    NOP
20641 124342  005737  003032            TST     CCHPAS                  ;have done enough inclusive passes?
20642 124346  001003                    BNE     99$                     ; not yet
20643 124350  000240                    NOP                             ; debug aid
20644 124352  000137  125304            JMP     ALLEND                  ; yes skip this
20645 124356  000240            99$:    NOP
20646
20647 124360  032737  000200  000052        BIT     #BIT07,@#52             ;UFD MODE?
20648 124366  001402                        BEQ     1$                      ;IF NOT, BRANCH
20649 124370  000137  140340                JMP     $EOP                    ;OTHERWISE, NEXT PASS
20650 124374  005737  002664        1$:     TST     CSR1                    ;AT LEAST 1 Q22BE?
20651 124400  001002                        BNE     2$                      ;IF YES, BRANCH
20652 124402  000137  140340                JMP     $EOP                    ;OTHERWISE, NEXT PASS
20653                                 ;
20654                                 ; TRY TO DO DMA WRITE AT ALL DIFFERENT PRIORITIES
20655                                 ;
20656 124406  012702  000340        2$:     MOV     #340,R2                 ;START WITH 7
20657 124412  000402                        BR      4$                      ;GO DO IT
20658 124414  162702  000040        3$:     SUB     #40,R2                  ;LOWER PRIORITY
20659 124420  005037  002740        4$:     CLR     TEMP                    ;CLEAR TEST LOCATION
20660 124424  005000                        CLR     R0                      ;DO JUST 1 WORD
20661 124426  106402                        MTPS    R2                      ;LOWER PRIORITY
20662 124430  004737  137370                JSR     PC,DMATRN               ;DO DATO
20663 124434  042737  001000  177520        BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
20664 124442  005737  002740                TST     TEMP                    ;STILL CACHED?
20665 124446  013737  177752  114150        MOV     HITMIS,RECDAT           ;STORE HIT/MISS
20666 124454  052737  001000  177520        BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
20667 124462  032737  000004  114150        BIT     #BIT02,RECDAT           ;LAST ACCESS HIT?
20668 124470  001401                        BEQ     5$                      ;IF MISS, BRANCH
20669 124472  104120                        ERROR   .120
20670 124474  022737  012525  002740  5$:   CMP     #12525,TEMP             ;DATO OK?
20671 124502  001401                        BEQ     6$                      ;IF SO, BRANCH
20672 124504  104123                        ERROR   .123                    ;DATO
```

TEST - DMA WRITE HIT CYCLES

```
20673 124506  005702                  6$:     TST     R2                      ;LAST PRIORITY 0?
20674 124510  001341                          BNE     3$                      ;IF NOT, BRANCH
20675 124512  106427  000340                  MTPS    #340                    ;RESTORE PRIORITY
20676                                 ;
20677                                 ;
20678                                 ; VERIFY THAT BYPASS WITH DMA DATI INVALIDATES CACHE
20679                                 ;
20680 124516  012737  001000  177746  8$:     MOV     #BIT09,CCR              ;SET BYPASS
20681 124524  004737  137370                  JSR     PC,DMATRN               ;DO DMA DATI
20682 124530  005037  177746                  CLR     CCR                     ;CLEAR BYPASS
20683 124534  042737  001000  177520          BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
20684 124542  005737  002740                  TST     TEMP                    ;IN CACHE?
20685 124546  013737  177752  114150          MOV     HITMIS,RECDAT           ;STORE REGISTER
20686 124554  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
20687 124562  032737  000004  114150          BIT     #BIT02,RECDAT           ;TEMP WAS A HIT?
20688 124570  001401                          BEQ     DATBO                   ;IF NOT, BRANCH
20689 124572  104123                          ERROR   +123
20690                                 ;
20691                                 ; DO DATBO
20692                                 ;
20693 124574  012701  000010          DATBO:  MOV     #10,R1                  ;COUNTER FOR 8
20694 124600  012702  002740                  MOV     #TEMP,R2                ;START WITH TEMP
20695 124604  005022                  1$:     CLR     (R2)+                   ;CLEAR ALL 16
20696 124606  077102                          SOB     R1,1$                   ;DO ALL 16
20697 124610  005200                          INC     R0                      ;FLAG BLOCK MODE
20698 124612  012777  002740  056050          MOV     #TEMP,@BA               ;LOAD DMA ADDRESS
20699 124620  012777  177770  056044          MOV     #177770,@WC             ;DO 8 WORDS
20700 124626  012777  001701  056030          MOV     #1701,@CSR1             ;16 WORDS FROM 32K
20701 124634  004737  137370                  JSR     PC,DMATRN               ;DO DATBO
20702 124640  032777  010000  056020          BIT     #BIT12,@CSR2            ;NO BLOCK MODE SLAVE?
20703 124646  001401                          BEQ     2$                      ;IF NOT, BRANCH
20704 124650  104123                          ERROR   +123                    ;NO BLOCK MODE SLAVE
20705 124652  012701  000004          2$:     MOV     #4,R1                   ;COUNTER FOR 4 LOCATIONS
20706 124656  012702  002740                  MOV     #TEMP,R2                ;START WITH TEMP
20707 124662  042737  001000  177520  3$:     BIC     #1000,BCSR              ;DISABLE HALT ON BREAK
20708 124670  005712                          TST     (R2)                    ;ACCESS A LOCATION
20709 124672  013737  177752  114150          MOV     HITMIS,RECDAT           ;STORE REGISTER
20710 124700  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
20711 124706  032737  000004  114150          BIT     #BIT02,RECDAT           ;HIT?
20712 124714  001401                          BEQ     4$                      ;IF NOT, BRANCH
20713 124716  104121                          ERROR   +121                    ;DMA DOES NOT INVALIDATE
20714 124720  022722  012525          4$:     CMP     #12525,(R2)+            ;DATO OK?
20715 124724  001401                          BEQ     5$                      ;IF SO,BRANCH
20716 124726  104123                          ERROR   +123                    ;IN DMA
20717 124730  062702  000002          5$:     ADD     #2,R2                   ;DO IN 2 WORDS
20718 124734  077126                          SOB     R1,3$                   ;DO ALL 16
20719
20720                                 ;
20721                                 ; DO 4K OF DATO AND CHECK FOR MISS
20722                                 ;
20723 124736  004737  136574                  JSR     PC,INITMM               ;SET UP MMU
20724 124742  012737  002000  172354          MOV     #2000,KIPAR6            ;START AT 32K
20725 124750  042737  100000  172314          BIC     #BIT15,KIPDR6           ;NO BYPASS
20726 124756  052737  000001  177572          BIS     #BIT00,MMR0             ;ENABLE MMU
20727 124764  012737  125124  000004          MOV     #DATI,@#4               ;IF 32K NXM
20728 124772  012702  140000                  MOV     #140000,R2              ;START WITH 32K
20729 124776  005712                          TST     (R2)                    ;EXIT
```

```
20730 125000  012701  010000          MOV     #10000,R1       ;COUNTER FOR 4K
20731 125004  005022            6$:   CLR     (R2)+           ;CLEAR ALL 16
20732 125006  077102                  SOB     R1,6$           ;DO ALL 16
20733 125010  005200                  INC     R0              ;FLAG BLOCK MODE
20734 125012  012777  000000  055650  MOV     #0,@BA          ;LOAD DMA ADDRESS
20735 125020  012777  170000  055644  MOV     #-10000,@WC     ;DO 4K
20736 125026  012777  003701  055630  MOV     #3701,@CSR1     ;16 WORDS FROM 32K
20737 125034  004737  137370          JSR     PC,DMATRN       ;DO DATBO
20738 125040  012701  004000    7$:   MOV     #4000,R1        ;COUNTER FOR 4K LOCATIONS
20739 125044  012702  140000          MOV     #140000,R2      ;START WITH 0
20740 125050  042737  001000  177520  8$:   BIC     #1000,BCSR      ;DISABLE HALT ON BREAK
20741 125056  005712                  TST     (R2)            ;ACCESS A LOCATION
20742 125060  013737  177752  114150  MOV     HITMIS,RECDAT   ;STORE REGISTER
20743 125066  052737  001000  177520  BIS     #1000,BCSR      ;ENABLE HALT ON BREAK
20744 125074  032737  000004  114150  BIT     #BIT02,RECDAT   ;HIT?
20745 125102  001401                  BEQ     9$              ;IF NOT, BRANCH
20746 125104  104121                  ERROR   +121            ;DMA DOES NOT INVALIDATE
20747 125106  022722  012525    9$:   CMP     #12525,(R2)+    ;DATO OK?
20748 125112  001401                  BEQ     10$             ;IF SO,BRANCH
20749 125114  104123                  ERROR   +123            ;IN DMA
20750 125116  062702  000002    10$:  ADD     #2,R2           ;DO IN 2 WORDS
20751 125122  077126                  SOB     R1,8$           ;DO ALL OF THEM
20752                           ;
20753                           ; CHECK DATI
20754                           ;
20755 125124  005037  177572    DATI: CLR     MMR0            ;DISABLE MMU
20756 125130  012706  001100          MOV     #1100,SP        ;RESTORE STACK
20757 125134  012737  137542  000004  MOV     #TOUT,@#4       ;AND TIMEOUT
20758 125142  012737  052525  002740  MOV     #52525,TEMP     ;LOAD MEMORY
20759 125150  005000                  CLR     R0              ;JUST 1 WORD
20760 125152  004737  137452          JSR     PC,DMARD        ;DO DATI
20761 125156  022777  052525  055510  CMP     #52525,@DATA    ;DATI OK?
20762 125164  001401                  BEQ     11$             ;IF YES, BRANCH
20763 125166  104123                  ERROR   +123            ;DATI
20764                           ;
20765                           ; DO DATBI
20766                           ;
20767 125170  012701  000010    11$:  MOV     #10,R1          ;COUNTER FOR 16 LOCATIONS
20768 125174  012702  002740          MOV     #TEMP,R2        ;START WITH TEMP
20769 125200  012703  002740          MOV     #TEMP,R3        ;START WITH TEMP
20770 125204  010322            12$:  MOV     R3,(R2)+        ;PUT ADDRESSES
20771 125206  005723                  TST     (R3)+           ;INCREMENT ADDRESS
20772 125210  077103                  SOB     R1,12$          ;DO ALL 16
20773 125212  005200                  INC     R0              ;FLAG BLOCK MODE
20774 125214  004737  137452          JSR     PC,DMARD        ;DO DATBI
20775 125220  032777  010000  055440  BIT     #BIT12,@CSR2    ;NO BLOCK MODE SLAVE?
20776 125226  001401                  BEQ     13$             ;IF NO, BRANCH
20777 125230  104123                  ERROR   +123
20778 125232  022777  002756  055434  13$:  CMP     #TEMP+16,@DATA  ;DATI OK?
20779 125240  001401                  BEQ     14$             ;IF SO, BRANCH
20780 125242  104123                  ERROR   +123            ;IN DATIB
20781 125244  042737  001000  177520  14$:  BIC     #1000,BCSR      ;DISABLE HALT ON BREAK
20782 125252  005737  002740          TST     TEMP            ;ACCESS
20783 125256  013737  177752  114150  MOV     HITMIS,RECDAT   ;STORE REGISTER
20784 125264  052737  001000  177520  BIS     #1000,BCSR      ;ENABLE HALT ON BREAK
20785 125272  032737  000004  114150  BIT     #BIT02,RECDAT   ;HIT?
20786 125300  001001                  BNE     15$             ;IF YES, BRANCH
```

TEST - DMA WRITE HIT CYCLES

```
 20787 125302  104123                          ERROR   +123
 20788 125304                        15$:
 20789 125304                        ALLEND:
```

```
                                .SBTTL  TEST  - DIFFERENT LEVELS OF INTERRUPTS
                                ;DIFFERENT LEVELS OF INTERRUPTS
                                ;THIS TEST WILL PROGRAM Q22 BUS EXERCISER TO INTERRUPT AT DIFFERENT
                                ;LEVELS.  ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS AND
                                ;PIRQ'S WILL BE TESTED.
                                ;
                                ;CHECK DIFFERENT LEVELS OF INTERRUPTS.
                                ;ROUTINE TEST
                                ;.      SET VECTOR TO INTERRUPT_DMA
                                ;.      FOR INTERRUPTS FROM 4 TO 7 DO
                                ;.      .       ENABLE INTERRUPTS
                                ;.      .       SET PRIORITY=INTERUPT
                                ;.      .       IF INTERRUPT_FLAG SET THEN
                                ;.      .       .       ERROR
                                ;.      .       ENDIF
                                ;.      .       ENABLE INTERRUPTS
                                ;.      .       SET PRIORITY=INTERRUPT-1
                                ;.      .       IF INTERRUPT_FLAG NOTSET THEN
                                ;.      .       .       ERROR
                                ;.      .       ENDIF
                                ;.      .       LET INTERRUPT_DMA=0
                                ;.      ENDDO
                                ;ENDROUTINE
                                ;
                                ;ROUTINE INTERUPT_DMA
                                ;.      LET INTERRUPT_FLAG=1
                                ;RETURN
                                ;ENDROUTINE

                                ;;**************************************************************************
 125304  000004                 TST62:  SCOPE
20821 125306  032737  000200  000052          BIT     #BIT07,@#52                     ;UFD MODE?
20822 125314  001122                          BNE     TST63           ;;IF SO, EXIT TEST
20823 125316  005737  002664                  TST     CSR1                            ;AT LEAST ONE Q22BE FOUND?
20824 125322  001517                          BEQ     TST63           ;;IF NOT, EXIT TEST
20825                                 ;
20826                                 ; SETUP INITIAL PRIORITY TO 7
20827                                 ;
20828 125324  013703  002664                  MOV     CSR1,R3                         ;DO FOR FIRST FOUND Q22BE
20829 125330  012777  125424  055340          MOV     #5$,@VQBE1                      ;POINT INTERRUPT VECTOR TO PROGRAM
20830 125336  012777  000340  055334          MOV     #340,@VQPR1                     ;AT PRIORITY 7
20831 125344  012700  003002                  MOV     #Q22EN,R0                       ;START WITH 7 FOR INTERRUPTS
20832 125350  012701  000340                  MOV     #340,R1                         ;LOW BOUNDARY FOR NO INTERRUPTS
20833 125354  000402                          BR      2$                              ;TRY TO DO IT FOR FIRST
20834 125356  162701  000040          1$:     SUB     #40,R1                          ;LOWER LOW BOUNDARY
20835                                 ;
20836                                 ; CHECK THAT INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN BR
20837                                 ;
20838 125362  012737  000340  001160  2$:     MOV     #340,$TMP0                      ;TOP PRIORITY FOR NO INTERRUPTS
20839 125370  000403                          BR      4$                              ;DO FIRST ONE
20840 125372  162737  000040  001160  3$:     SUB     #40,$TMP0                       ;DO AT NEXT LEVEL
20841 125400  106437  001160          4$:     MTPS    $TMP0                           ;SET PRIORITY NOT TO INTERRUPT
20842 125404  004737  137350                  JSR     PC,Q22INT                       ;ENABLE INTERRUPTS
20843 125410  012077  055252                  MOV     (R0)+,@CSR2                     ;CLEAR GO BIT
20844 125414  000240                          NOP
20845 125416  000240                          NOP
20846 125420  000240                          NOP
```

TEST - DIFFERENT LEVELS OF INTERRUPTS

```
20847 125422  000403                        BR      6$                      ;IF NO INTERUPT, BRANCH
20848 125424  104126              5$:       ERROR   +126                    ;INTERRUPTS HAPPEN
20849 125426  005726                        TST     (SP)+                   ;RESTORE STACK
20850 125430  005726                        TST     (SP)+
20851 125432  020137  001160        6$:     CMP     R1,$TMP0                ;LAST ONE?
20852 125436  001355                        BNE     3$                      ;IF NOT BRANCH
20853 125440  022710  000002                CMP     #2,(R0)                 ;AT BR4?
20854 125444  001344                        BNE     1$                      ;IF NOT LAST ONE, BRANCH
20855                                     ;
20856                                     ; INTERRUPT AT ALL LEVELS
20857                                     ;
20858 125446  012777  125542  055222  INQ22: MOV     #5$,@VQBE1              ;POINT INTERRUPT VECTOR TO PROGRAM
20859 125454  012777  000340  055216        MOV     #340,@VQPR1             ;AT PRIORITY 7
20860 125462  012700  003002                MOV     #Q22EN,R0               ;START WITH 7 FOR INTERRUPTS
20861 125466  012701  000300                MOV     #300,R1                 ;TOP BOUNDARY FOR INTERRUPTS
20862 125472  000402                        BR      2$                      ;TRY TO DO IT FOR FIRST
20863 125474  162701  000040        1$:     SUB     #40,R1                  ;LOWER TOP BOUNDARY
20864                                     ;
20865                                     ; CHECK THAT INTERRUPTS HAPPEN AT PRIORITY LOWER THAN BR
20866                                     ;
20867 125500  010137  001160        2$:     MOV     R1,$TMP0                ;PRIORITY FOR INTERRUPTS
20868 125504  000403                        BR      4$                      ;DO FIRST ONE
20869 125506  162737  000040  001160  3$:     SUB     #40,$TMP0               ;DO AT NEXT LEVEL
20870 125514  106437  001160        4$:     MTPS    $TMP0                   ;SET PRIORITY NOT TO INTERRUPT
20871 125520  004737  137350                JSR     PC,Q22INT               ;ENABLE INTERRUPTS
20872 125524  011077  055136                MOV     (R0),@CSR2              ;CLEAR GO BIT
20873 125530  000240                        NOP                             ;WAIT A WHILE
20874 125532  000240                        NOP
20875 125534  000240                        NOP
20876 125536  104126                        ERROR   +126                    ;INTERRUPTS DON'T HAPPEN
20877 125540  000402                        BR      6$                      ;DON'T RESTORE STACK
20878 125542  005726              5$:   .   TST     (SP)+                   ;RESTORE STACK
20879 125544  005726                        TST     (SP)+
20880 125546  005737  001160        6$:     TST     $TMP0                   ;LAST ONE 0?
20881 125552  001355                        BNE     3$                      ;IF NOT BRANCH
20882 125554  022720  000002                CMP     #2,(R0)+                ;AT BR4?
20883 125560  001345                        BNE     1$                      ;IF NOT LAST ONE, BRANCH
20884
20885
```

```
                                .SBTTL  TEST  - ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS
                                ;CHECK PRIORITY ORDER BETWEEN PIRQ'S AND INTERRUPTS.
                                ;ROUTINE TEST
                                ;.      IF UFD THEN
                                ;.      .       EXIT TEST
                                ;.      ENDIF
                                ;.      DO FOR I FROM #6 DOWN TO #3
                                ;.      .       SET PRIORITY TO I
                                ;.      .       ENABLE INTERRUPT(I+1) AND PIRQ(I+1)
                                ;.      .       IF INTERRUPT(I+1) WAS BEFORE PIRQ(I+1) THEN
                                ;.      .       .       ERROR
                                ;.      .       ENDIF
                                ;.      ENDDO
                                ;ENDROUTINE
                                ;;*****************************************************************
    125562  000004              TST63:  SCOPE
20903 125564  032737  000200  000052        BIT     #BIT07,@#52                     ;UFD MODE?
20904 125572  001065                        BNE     TST64           ;;IF SO, EXIT TEST
20905 125574  005737  002664                TST     CSR1                            ;AT LEAST ONE Q22BE FOUND?
20906 125600  001462                        BEQ     TST64           ;;IF NOT, EXIT TEST
20907 125602  012777  125710  055066        MOV     #3$,@VQBE1                      ;SETUP Q22BE VECTOR
20908 125610  012777  000340  055062        MOV     #340,@VQPR1                     ;AT PRIORITY 7
20909 125616  012737  125720  000240        MOV     #4$,PIRQVEC                     ;SETUP PIRQ VECTOR
20910 125624  012737  000340  000242        MOV     #340,PIRQVEC+2                  ;AT PRIORITY 7
20911 125632  012700  003002                MOV     #Q22EN,R0                       ;POINT THRU PRIORITIES FOR Q22BE
20912 125636  012704  125736                MOV     #PIRQT,R4                       ;POINTER THRU PIRQ'S
20913 125642  013703  002664                MOV     CSR1,R3                         ;DO FOR FIRST Q22BE
20914 125646  012702  000300                MOV     #300,R2                         ;START WITH CPU PRIORITY AT 7
20915 125652  000402                        BR      2$                              ;DO FIRST ONE
20916 125654  162702  000040          1$:   SUB     #40,R2                          ;LOWER CPU PRIORITY
20917 125660  106427  000340          2$:   MTPS    #340                            ;RAISE PRIORITY TO 7
20918 125664  012437  177772                MOV     (R4)+,PIRQ                      ;SET PRIORITY FOR PIRQ'S
20919 125670  004737  137350                JSR     PC,Q22INT                       ;INITIALISE Q22BE TO INTERRUPT
20920 125674  012077  054766                MOV     (R0)+,@CSR2                     ;SET DONE BIT
20921 125700  106402                        MTPS    R2                              ;LOWER PRIORITY
20922 125702  000240                        NOP
20923 125704  000240                        NOP
20924 125706  000240                        NOP
20925 125710  104124          3$:   ERROR   +124                                    ;PIRQ'S DON'T TAKE OVER BIRQ'S
20926 125712  005726                        TST     (SP)+                           ;CLEAN UP STACK
20927 125714  005726                        TST     (SP)+
20928 125716  000402                        BR      5$                              ;BRANCH AROUND PIRQ INTERRUPT
20929 125720  005726          4$:   TST     (SP)+                                   ;CLEAN UP STACK
20930 125722  005726                        TST     (SP)+
20931 125724  022702  000140          5$:   CMP     #140,R2                         ;PRIORITY 3 LAST ONE?
20932 125730  001351                        BNE     1$                              ;IF NOT BRANCH
20933 125732  005037  177772                CLR     PIRQ                            ;CLEAR ANY REQUESTS
20934
20935 125736  100000  040000  020000  PIRQT: .WORD  100000,40000,20000,10000        ;PIRQ'S<7-4>
20936
```

```
                                        .SBTTL  TEST    - POWER DOWN TEST
                                        ;USING Q22BE THIS TEST WILL CHECK THAT ON POWER DOWN CONDITION IF
                                        ;POWER UP CODE 00 IS SELECTED THE CPU TRAPS THRU 24
                                        ;ROUTINE TEST
                                        ;.      IF UFD OR POWER UP CODE 00 NOT SELECTED THEN
                                        ;.      .       EXIT TEST
                                        ;.      ENDIF
                                        ;.      SET 24 TO POINT TO TEST AREA
                                        ;.      LET CSR2<5> = #1 TO NEGATE BPOK
                                        ;.      IF NO TRAP TO 24 THEN
                                        ;.      .       ERROR IN POWER DOWN CYCLE
                                        ;.      ENDIF
                                        ;.      IF TRAP TO 24 THEN
                                        ;.      .       LET CSR2<5> = #0
                                        ;.      ENDIF
                                        ;ENDROUTINE

                                        ;;*****************************************************************
     125746  000004                     TST64:  SCOPE
20956 125750  032737  000200  000052          BIT     #BIT07,@#52                             ;UFD MODE?
20957 125756  001033                          BNE     TST65           ;;EXIT TEST IN UFD  MODE
20958 125760  005737  002664                  TST     CSR1                                    ;AT LEAST ONE Q22BE FOUND?
20959 125764  001430                          BEQ     TST65           ;;IF NOT, EXIT TEST
20960 125766  013737  000024  001160          MOV     PWRVEC,$TMP0                            ;SAVE POWER UP VECTOR
20961 125774  012737  126030  000024          MOV     #1$,PWRVEC                              ;POINT NEW TO PROGRAM
20962 126002  012737  000340  000026          MOV     #340,PWRVEC+2                           ;AT PRIORITY 7
20963 126010  012777  000040  054650          MOV     #BIT05,@CSR2                            ;DO POWER DOWN
20964 126016  000240                          NOP
20965 126020  005077  054642                  CLR     @CSR2                                   ;CLEAR POWER DOWN BIT
20966 126024  104125                          ERROR   +125                                    ;NO POWER DOWN TRAP
20967 126026  000404                          BR      2$                                      ;SKIP RESTORING STACK
20968 126030  005077  054632          1$:     CLR     @CSR2                                   ;CLEAR POWER DOWN BIT
20969 126034  005726                          TST     (SP)+                                   ;RESTORE STACK POINTER
20970 126036  005726                          TST     (SP)+
20971 126040  013737  001160  000024  2$:     MOV     $TMP0,PWRVEC                            ;RESTORE POWER VECTOR
20972
20973
20974
```

TEST - ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS

```
20976                                   .SBTTL TEST    - ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS
20977                                   ;IF TWO Q22BE ARE AVAILABLE, THIS TEST WILL CHECK THAT HIGHER LEVEL
20978                                   ;INTERRUPT REQUESTS TAKE PRIORITY OVER LOWER LEVEL ONES
20979                                   ;ROUTINE TEST
20980                                   ;.      IF UFD OR 2ND Q22BE NOT AVAILABLE THEN
20981                                   ;.      .       EXIT TEST
20982                                   ;.      ENDIF
20983                                   ;.      DO FOR PRIORITY_LEVELS FROM 4 TO 6
20984                                   ;.      .       SET UP 1ST Q22BE TO INTERRUPT AT PRIORITY_LEVEL
20985                                   ;.      .       SET UP 2ND Q22BE TO INTERRUPT AT PRIORITY_LEVEL+1
20986                                   ;.      .       SET UP CPU PRIORITY TO PRIORITY_LEVEL-1
20987                                   ;.      .       IF INTERRUPTS FORM 1ST Q22BE HAPPENED BEFORE 1ST THEN
20988                                   ;.      .       .       ERROR
20989                                   ;.      .       ENDIF
20990                                   ;.      ENDDO
20991                                   ;ENDROUTINE
20992
20993                                   ;;*****************************************************************
     126046  000004                     TST65:  SCOPE
20994 126050  032737  000200  000052            BIT     #BIT07,@#52                     ;UFD MODE?
20995 126056  001064                     BNE     TST66           ;;IF SO, EXIT TEST
20996 126060  005737  002704                     TST     CSR12                           ;SECOND Q22BE AVAILABLE?
20997 126064  001002                     BNE     1$                                      ;IF YES, GO DO TEST
20998 126066  000137  140340                     JMP     $EOP                            ;OTHERWISE, GOTO EOP
20999                                   ;
21000                                   ; INITIALISE VECTORS FOR Q22BE'S
21001                                   ;
21002 126072  012777  126206  054576    1$:     MOV     #4$,@VQBE1                      ;VECTOR FOR 1ST ONE
21003 126100  012777  000340  054572            MOV     #340,@VQPR1                     ;AT PRIORITY 7
21004 126106  012777  126216  054602            MOV     #5$,@VQBE2                      ;VECTOR FOR 2ND ONE
21005 126114  012777  000340  054576            MOV     #340,@VQPR2                     ;AT PRIORITY 7
21006 126122  012700  003004                     MOV     #Q22EN+2,R0                     ;POINTER FOR Q22BE BR'S
21007 126126  012702  000240                     MOV     #240,R2                         ;CPU AT 5
21008 126132  000402                     BR      3$                              ;START DOING ARBITRATION
21009                                   ;
21010                                   ; DO FOR CPU PRIORITIES 5-3
21011                                   ;
21012 126134  162702  000040            2$:     SUB     #40,R2                          ;LOWER CPU PRIORITY
21013 126140  106427  000340            3$:     MTPS    #340                            ;CPU AT 7
21014 126144  013703  002704                     MOV     CSR12,R3                        ;Q22BE 2 AT HIGHER BR
21015 126150  004737  137350                     JSR     PC,Q22INT                       ;INITIALISE TO INTERRUPTS
21016 126154  011077  054506                     MOV     (R0),@CSR2                      ;START 1ST ONE
21017 126160  013703  002664                     MOV     CSR1,R3                         ;Q22BE 1 AT LOWER BR
21018 126164  005740                     TST     -(R0)                           ;HIGHER PRIORITY
21019 126166  004737  137350                     JSR     PC,Q22INT                       ;INITIALISE
21020 126172  012077  054510                     MOV     (R0)+,@CSR22                    ;START 2ND ONE
21021 126176  106402                     MTPS    R2                              ;LOWER CPU PRIORITY
21022 126200  000240                     NOP                                     ;WAIT A WHILE
21023 126202  000240                     NOP
21024 126204  000240                     NOP
21025 126206  104126            4$:     ERROR   +126                            ;INTERRUPTS IN WRONG ORDER
21026 126210  005726                     TST     (SP)+                           ;RESTORE STACK
21027 126212  005726                     TST     (SP)+
21028 126214  000402                     BR      6$
21029 126216  005726            5$:     TST     (SP)+                           ;RESTORE STACK
21030 126220  005726                     TST     (SP)+
21031 126222  022702  000140            6$:     CMP     #140,R2                         ;PRIORITY 3 LAST?
```

TEST - ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS SEQ 0393

```
21032 126226  001342                    BNE     2$                    ;IF NOT, BRANCH TO CONTINUE
21033
```

```
21035                                   .SBTTL  TEST   - PMG COUNTER
21036                                   ;USING 2 Q22BE THIS TEST WILL VALIDATE THAT PMG COUNTER IS REALLY
21037                                   ;CAPABLE OF GRANTING CPU BUS MASTERSHIP WHEN DMA REQUESTS ARE STILL
21038                                   ;PENDING.
21039                                   ;ROUTINE TEST
21040                                   ;.      IF UFD OR NO 2ND Q22BE THEN
21041                                   ;.      .       EXIT TEST
21042                                   ;.      ENDIF
21043                                   ;.      SET PMG COUNT TO SOME VALUE
21044                                   ;.      PROGRAM BOTH Q22BE TO INTERRUPT AT THE SAME LEVEL
21045                                   ;.      DETERMINE WHICH HAS HIGHER PRIORITY ON THE BUS
21046                                   ;.      PROGRAM Q22BE WITH HIGHER PRIORITY TO DO HOG MODE
21047                                   ;.      PROGRAM 2ND ONE TO DO A CYCLE
21048                                   ;.      CACHE INSTRUCTION THAT WILL STOP 2ND DMA
21049                                   ;.      INITIATE BOTH DMA CYCLES
21050                                   ;.      STOP 2ND DMA (RESET IS "STOLEN" CPU BUS CYCLE)
21051                                   ;.      IF 2ND DMA HAPPENED THEN
21052                                   ;.      .       ERROR
21053                                   ;.      ENDIF
21054                                   ;.      CLEAR PMG COUNT
21055                                   ;.      PROGRAM Q22BE WITH HIGHER PRIORITY TO DO HOG MODE
21056                                   ;.      PROGRAM 2ND ONE TO DO A CYCLE
21057                                   ;.      CACHE INSTRUCTION THAT WILL STOP 2ND DMA
21058                                   ;.      INITIATE BOTH DMA CYCLES
21059                                   ;.      STOP 2ND DMA (CLEARING OF CSR2<0> IS "STOLEN" CPU BUS CYCLE)
21060                                   ;.      IF 2ND DMA DIDN'T HAPPENED THEN
21061                                   ;.      .       ERROR
21062                                   ;.      ENDIF
21063                                   ;ENDROUTINE
21064
21065                                   ;;*****************************************************************
     126230  000004                   TST66:  SCOPE
21066 126232  000240                           NOP
21067 126234  005737  003032                   TST     CCHPAS                  ;have done enough inclusive passes?
21068 126240  001003                           BNE     99$                     ; not yet
21069 126242  000240                           NOP                             ; debug aid
21070 126244  000137  126750                   JMP     11$                     ; yes skip this
21071 126250  000240                   99$:    NOP
21072
21073 126252  032737  000200  000052           BIT     #BIT07,@#52                     ;UFD MODE?
21074 126260  001402                           BEQ     100$                            ;IF NOT, CONTINUE
21075 126262  000137  140340                   JMP     $EOP                            ;EXIT
21076 126266  005737  002704           100$:   TST     CSR12                           ;SECOND Q22BE AVAILABLE?
21077 126272  001002                           BNE     110$                            ;IF YES, GO DO TEST
21078 126274  000137  140340                   JMP     $EOP                            ;OTHERWISE, GOTO EOP
21079                                   ;
21080                                   ; DETERMINE WHICH Q22BE IS CLOSER TO CPU BY DOING DATO
21081                                   ; THE CSR1 OF THE ONE WITH HIGHER PRIORITY IS IN R4, THE SECOND IN R2
21082                                   ;
21083 126300  005077  054362           110$:   CLR     @CSR2                           ;CLEAR JUST IN CASE
21084 126304  005077  054376                   CLR     @CSR22
21085 126310  012777  002740  054352           MOV     #TEMP,@BA                       ;ADDRESS TO BE USED
21086 126316  012777  002740  054364           MOV     #TEMP,@BA2                      ;FOR BOTH OF THEM
21087 126324  012777  177777  054340           MOV     #177777,@WC                     ;DO FOR 1 WORD
21088 126332  012777  177777  054352           MOV     #177777,@WC2                    ;IN BOTH Q22BW'S
21089 126340  012777  012525  054326           MOV     #12525,@DATA                    ;DATA FOR 1ST
21090 126346  012777  052525  054340           MOV     #52525,@DATA2                   ;DATA FOR 2ND
```

```
21091 126354  012777  001601  054302          MOV     #1601,@CSR1                     ;1 DATO FOR 1ST
21092 126362  012777  001601  054314          MOV     #1601,@CSR12                    ;AND 2ND
21093 126370  012777  000001  054304          MOV     #1,@SIMGOA                      ;BOTH GO
21094 126376  105777  054262          1$:     TSTB    @CSR1                           ;FIRST DONE?
21095 126402  100375                          BPL     1$                              ;IF NOT, WAIT
21096 126404  105777  054274          2$:     TSTB    @CSR12                          ;SECOND DONE
21097 126410  100375                          BPL     2$                              ;WAIT FOR 2ND
21098 126412  022737  052525  002740          CMP     #52525,TEMP                     ;SECOND FINISHED LAST?
21099 126420  001405                          BEQ     3$                              ;IF SO, BRANCH
21100 126422  013704  002704                  MOV     CSR12,R4                        ;SECOND ONE AT HIGHER LEVEL
21101 126426  013702  002664                  MOV     CSR1,R2                         ;FIRST AT LOWER
21102 126432  000404                          BR      4$                              ;DO PMG PART
21103 126434  013704  002664          3$:     MOV     CSR1,R4                         ;FIRST ONE AT HIGHER LEVEL
21104 126440  013702  002704                  MOV     CSR12,R2                        ;SECOND AT LOWER
21105                                 ;
21106                                 ; INITIATE DMA CYCLES TO WORK WITH PMG COUNTER
21107                                 ;
21108 126444  013737  177520  002730  4$:     MOV     BCSR,SAVBR                      ;STORE BCSR REGISTER
21109 126452  012700  000001                  MOV     #BIT00,R0                       ;FIRST VALUE FOR PMG 0.4msec
21110 126456  050037  177520          5$:     BIS     R0,BCSR                         ;CHANGE PMG CONUTER
21111 126462  005737  126564                  TST     6$                              ;CACHE RESET INTRUCTION
21112 126466  005737  126566                  TST     6$+2                            ;AND THE FOLLOWING FEW
21113 126472  005737  126570                  TST     6$+4                            ;
21114 126476  005737  126572                  TST     6$+6                            ;
21115 126502  012714  001407                  MOV     #1407,(R4)                      ;DATI IN HOG MODE FOR HIGHER ONE
21116 126506  005064  000002                  CLR     2(R4)                           ;CLEAR CSR2
21117 126512  012764  002740  000004          MOV     #TEMP,4(R4)                     ;ADDRESS TO START DATI
21118 126520  012764  000000  000006          MOV     #0,6(R4)                        ;DO 128 DATI'S IN HOG MODE
21119 126526  012712  001407                  MOV     #1407,(R2)                      ;DATO FOR LOWER LEVEL ONE
21120 126532  005062  000002                  CLR     2(R2)                           ;CLEAR JUST IN CASE
21121 126536  012762  002740  000004          MOV     #TEMP,4(R2)                     ;USE THE SAME ADDRESS
21122 126544  012762  177777  000006          MOV     #177777,6(R2)                   ;DO JUST ONE DATO
21123 126552  005062  000010                  CLR     10(R2)                          ;CLEAR DATA OF 2ND Q22BE
21124 126556  012762  000001  000016          MOV     #1,16(R2)                       ;BOTH GO
21125 126564  000005                  6$:     RESET                                   ;STOP Q22BE AT R4
21126 126566  023762  002740  000010          CMP     TEMP,10(R2)                     ;SECOND DMA DONE?
21127 126574  001001                          BNE     7$                              ;IF SET, BRANCH
21128 126576  104127                          ERROR   +127                            ;NO CYCLE STEALING
21129 126600  062700  000003          7$:     ADD     #3,R0                           ;DO FOR 1,3,7 IN PMG
21130 126604  040037  177520                  BIC     R0,BCSR                         ;CLEAR PREVOIUS BITS IN BCSR
21131 126610  022700  000007                  CMP     #7,R0                           ;LAST ONE?
21132 126614  002320                          BGE     5$                              ;IF NOT, BRANCH
21133                                 ;
21134                                 ; TRY WITHOUT PMG COUNTER OPERATING
21135                                 ;
21136 126616  042737  000007  177520          BIC     #7,BCSR                         ;TURN OF PMG COUNTER
21137 126624  005737  126726                  TST     8$                              ;CACHE RESET FETCH
21138 126630  005737  126730                  TST     8$+2                            ;AND THE FOLLOWING FEW
21139 126634  005737  126732                  TST     8$+4                            ;
21140 126640  005737  126734                  TST     8$+6                            ;
21141 126644  012714  001407                  MOV     #1407,(R4)                      ;DATI IN HOG MODE FOR HIGHER ONE
21142 126650  005064  000002                  CLR     2(R4)                           ;CLEAR CSR2
21143 126654  012764  002740  000004          MOV     #TEMP,4(R4)                     ;ADDRESS TO START DATI
21144 126662  012764  000000  000006          MOV     #0,6(R4)                        ;DO 128 DATI'S IN HOG MODE
21145 126670  012712  001407                  MOV     #1407,(R2)                      ;DATO FOR LOWER LEVEL ONE
21146 126674  005062  000002                  CLR     2(R2)                           ;CLEAR CRS2 OF 2ND
21147 126700  012762  002740  000004          MOV     #TEMP,4(R2)                     ;USE THE SAME ADDRESS
```

TEST - PMG COUNTER

```
21148 126706  012762  177777  000006          MOV     #177777,6(R2)                   ;DO JUST ONE DATO
21149 126714  005062  000010                  CLR     10(R2)                          ;CLEAR DATA OF 2ND Q22BE
21150 126720  012777  000001  053754          MOV     #1,@SIMGOA                      ;BOTH GO
21151 126726  000005                  8$:     RESET                                   ;IF NOT WORKING, STOPS 2ND
21152 126730  023762  002740  000010          CMP     TEMP,10(R2)                     ;2ND DMA HAPPENED?
21153 126736  001401                          BEQ     9$                              ;IF YES, BRANCH
21154 126740  104127                          ERROR   +127                            ;IN PMG COUNTER
21155 126742  013737  002730  177520  9$:     MOV     SAVBR,BCSR                      ;RESTORE BCSR
21156 126750  000240                  11$:    NOP
21157 126752  000137  140340                  jmp     $eop
21158
21159 126756  123727  001220  000001  VIREOP: CMPB    $ENV,#1                         ; if not APT, don't worry about
21160 126764  001005                          BNE     1$                              ;
21161 126766  005737  003032                  TST     CCHPAS                          ; maintain cache routin pascnt
21162 126772  001402                          BEQ     1$
21163 126774  005337  003032                  DEC     CCHPAS
21164
21165                                         ; This VIREOP ROUTINE to provide common End of Pass exit point
21166 127000  000205                  1$:     rts     r5
21167
```

GLOBAL ERROR MESSAGES

```
21169                                   .SBTTL  GLOBAL ERROR MESSAGES
21170 127002     102     101     123    EM1:    .ASCIZ  /BASIC INSTRUCTION SET ERROR/
21171 127036     115     115     125    EM2:    .ASCIZ  /MMU ERROR/
21172 127050     106     120     120    EM3:    .ASCIZ  /FPP ERROR/
21173 127062     105     122     122    EM4:    .ASCIZ  /ERROR IN READ-WRITE BITS OF CCR/
21174 127122     106     117     122    EM5:    .ASCIZ  /FORCE MISS WRITES TO CACHE/
21175 127155     106     117     122    EM6:    .ASCIZ  /FORCE MISS WRITE INVALIDATES CACHE/
21176 127220     125     116     105    EM7:    .ASCIZ  /UNEXPECTED PARITY INTERRUPT/
21177 127254     124     101     107    EM10:   .ASCIZ  /TAG PARITY ERROR/
21178 127275     104     101     124    EM11:   .ASCIZ  /DATA PARITY ERROR/
21179 127317     114     117     127    EM12:   .ASCIZ  /LOW BYTE PARITY ERROR/
21180 127345     110     111     107    EM13:   .ASCIZ  /HIGH BYTE PARITY ERROR/
21181 127374     105     122     122    EM14:   .ASCIZ  /ERROR IN DATA PATH/
21182 127417     106     117     122    EM15:   .ASCIZ  /FORCE MISS READS FROM CACHE/
21183 127453     106     117     122    EM16:   .ASCIZ  /FORCE MISS READS FROM CACHE AND MISS/
21184 127520     105     122     122    EM17:   .ASCIZ  \ERROR IN RECORDING HITS IN HIT/MISS\
21185 127564     127     122     111    EM20:   .ASCIZ  /WRITE BYTE ALLOCATES CACHE/
21186 127617     127     122     111    EM21:   .ASCIZ  /WRITE BYTE HIT DOES NOT RECORD HIT/
21187 127662     102     131     124    EM22:   .ASCIZ  /BYTES REVERSED ON WRITE CYCLES/
21188 127721     103     117     116    EM23:   .ASCIZ  /CONDITIONAL BYPASS DOESN'T INVALIDATE CACHE/
21189 127775     110     111     124    EM24:   .ASCIZ  /HITS RECORDED AFTER FLUSHING CACHE/
21190 130040     102     131     120    EM25:   .ASCIZ  /BYPASS DOESN'T INVALIDATE CACHE/
21191 130100     115     123     105    EM26:   .ASCIZ  /MSER DOES NOT CLEAR ON WRITE REFERENCE/
21192 130147     120     101     122    EM27:   .ASCIZ  /PARITY ERROR DON'T CAUSE A MISS/
21193 130207     120     101     122    EM30:   .ASCIZ  /PARITY ERROR DON'T SET MSER WITH CCR<7>=0/
21194 130261     120     101     122    EM31:   .ASCIZ  /PARITY ERROR IGNORED/
21195 130306     120     101     122    EM32:   .ASCIZ  /PARITY ERROR IGNORED ON LOW BYTE/
21196 130347     120     101     122    EM33:   .ASCIZ  /PARITY ERROR IGNORED ON HIGH BYTE/
21197 130411     120     101     122    EM34:   .ASCIZ  /PARITY ABORT LOGIC DOESN'T WORK/
21198 130451     115     123     105    EM35:   .ASCIZ  /MSER NOT SET PROPERLY/
21199 130477     120     101     122    EM36:   .ASCIZ  /PARITY INTERRUPT LOGIC DOESN'T WORK/
21200 130543     116     130     115    EM37:   .ASCIZ  /NXM AND PARITY ABORT DIN'T HAPPEN/
21201 130605     120     101     122    EM40:   .ASCIZ  /PARITY ABORT NOT BLOCKED BY NXM TRAP/
21202 130652     115     125     114    EM41:   .ASCIZ  /MULTI-PROCESSOR HOOK INSTRUCTION DOESN'T CAUSE MISS/
21203 130736     105     122     122    EM42:   .ASCIZ  /ERROR IN PARITY LOGIC/
21204 130764     105     122     122    EM43:   .ASCIZ  /ERROR IN CACHE DATA RAMS/
21205 131015     105     122     122    EM44:   .ASCIZ  /ERROR IN NXM IN STANDALONE MODE/
21206 131055     105     122     122    EM45:   .ASCIZ  \ERROR IN RECORDING HITS THROUGH HIT/MISS REGISTER\
21207 131137     110     111     124    EM46:   .ASCIZ  /HIT RECORDED FOR A LOCATION THAT SHOULD NOT BE IN CACHE/
21208 131227     115     111     123    EM47:   .ASCIZ  /MISS RECORDED FOR A LOCATION THAT SHOULD BE IN CACHE/
21209 131314     105     122     122    EM50:   .ASCIZ  /ERROR IN TAG STORE/
21210 131337     105     122     122    EM51:   .ASCIZ  /ERROR PCR READ-WRITE BITS/
21211 131371     105     122     122    EM52:   .ASCIZ  /ERROR IN BCSR READ-WRITE BITS/
21212 131427     122     105     123    EM53:   .ASCIZ  /RESET DOESN'T CLEAR BCSR<4>/
21213 131463     103     110     105    EM54:   .ASCIZ  /CHECKSUM ERROR IN 16-BIT ROM /
21214 131521     103     110     105    EM55:   .ASCIZ  /CHECKSUM ERROR IN 8-BIT ROM/
21215 131555     124     111     115    EM56:   .ASCIZ  /TIMEOUT READING LKS/
21216 131601     114     113     123    EM57:   .ASCIZ  /LKS<07> DOES NOT BECOME 1/
21217 131633     127     122     111    EM60:   .ASCIZ  /WRITE REFERENCE DOESN'T CLEAR LKS<07>/
21218 131701     111     114     114    EM61:   .ASCIZ  /ILLEGAL LKS INTERRUPTS/
21219 131730     120     122     117    EM62:   .ASCIZ  /PROCESSOR INTERRUPTS DON'T CLEAR LKS<07>/
21220 132001     114     113     123    EM63:   .ASCIZ  /LKS READY DOESN'T GO LOW/
21221 132032     127     122     117    EM64:   .ASCIZ  /WRONG NUMBER OF LKS INTERRUPTS/
21222 132071     114     113     123    EM65:   .ASCIZ  /LKS INTERRUPTS HAPPEN AT WRONG PRIORITY/
21223 132141     102     103     123    EM66:   .ASCIZ  /BCSR<12> DOES NOT DISABLES LKS/
21224 132200     102     103     123    EM67:   .ASCIZ  /BCSR<13> DOESN'T SET LKS<06>/
21225 132235     122     105     123    EM70:   .ASCIZ  /RESET DOESN'T SET LKS<7>/
```

GLOBAL ERROR MESSAGES

```
21226 132266     122     105     123  EM71:   .ASCIZ  /RESET DOESN'T CLEAR LKS<06>/
21227 132322     124     111     115  EM72:   .ASCIZ  /TIMEOUT READING SLU REGISTERS/
21228 132360     105     122     122  EM73:   .ASCIZ  /ERROR IN XMIT READY/
21229 132404     122     103     123  EM74:   .ASCIZ  /RCSR<7> DOESN'T BECOME 1/
21230 132435     127     122     117  EM75:   .ASCIZ  /WRONG CHARACTER RECEIVED/
21231 132466     122     103     123  EM76:   .ASCIZ  /RCSR<07> NOT CLEARED AFTER READING RBUF/
21232 132536     130     103     123  EM77:   .ASCIZ  /XCSR<07> NOT SET ON RESET/
21233 132570     122     103     123  EM100:  .ASCIZ  /RCSR<07> NOT CLEARED ON RESET/
21234 132626     123     114     125  EM101:  .ASCIZ  /SLU INTERRUPTS HAPPEN AT 4/
21235 132661     122     105     123  EM102:  .ASCIZ  /RESET DOES NOT CLEAR PROPER BITS IN SLU REGISTERS/
21236 132743     124     122     101  EM103:  .ASCIZ  /TRANSMIT INTERRUPT DOES NOT CLEAR XCSR<07>/
21237 133016     122     105     103  EM104:  .ASCIZ  /RECEIVE INTERRUPTS DON'T CLEAR RCSR<07>/
21238 133066     102     122     105  EM105:  .ASCIZ  /BREAK CONDITION DOES NOT SET RBUF PROPERLY/
21239 133141     122     102     125  EM106:  .ASCIZ  /RBUF <15-11> WASN'T CLEARED ON NEXT CHARACTER/
21240 133217     123     114     125  EM107:  .ASCIZ  /SLU INTERRUPTS DON'T HAPPEN/
21241 133253     105     122     122  EM110:  .ASCIZ  /ERROR IN WRITING TO SLU REGISTERS/
21242 133315     106     111     122  EM111:  .ASCIZ  /FIRST CHARACTER WAS NOT OVERRUN BY THE SECOND/
21243 133373     117     126     105  EM112:  .ASCIZ  /OVERRUN CONDITION DOES NOT SET PROPER BITS IN RBUF/
21244 133456     117     126     105  EM113:  .ASCIZ  /OVERRUN BITS WERE NOT CLEARED ON THE NEXT CHARACTER/
21245 133542     105     122     122  EM114:  .ASCIZ  \ERROR ON XCSR<2>\
21246 133563     105     122     122  EM115:  .ASCIZ  /ERROR IN TAG STORE FROM STANDALONE MODE/
21247 133633     104     115     101  EM116:  .ASCIZ  /DMA TAG PARITY DOES NOT GENERATE PROPER RESPONSE/
21248 133714     115     123     105  EM117:  .ASCIZ  /MSER<13>NOT SET IN STANDALONE MODE/
21249 133757     104     115     101  EM120:  .ASCIZ  /DMA WRITE HITS DON'T INVALIDATE CACHE/
21250 134025     111     116     040  EM121:  .ASCIZ  /IN BLOCK MODE ON WRITE DMA HITS NOT EVERYTHING IS INVALIDATED/
21251 134123     122     105     101  EM122:  .ASCIZ  /READ DMA HIT IS WRONG/
21252 134151     105     122     122  EM123:  .ASCIZ  /ERROR IN Q22BE DMA CYCLES/
21253 134203     120     111     122  EM124:  .ASCIZ  /PIRQ INTERRUPTS DON'T TAKE PRIORITY OVER Q BUS INTERRUPTS/
21254 134275     116     117     040  EM125:  .ASCIZ  /NO POWER DOWN TRAP TO 24 OCCUR/
21255 134334     105     122     122  EM126:  .ASCIZ  /ERROR DOING Q22BE INTERRUPTS/
21256 134371     105     122     122  EM127:  .ASCIZ  /ERROR IN OPERATION OF PMG COUNTER/
21257 134433     125     116     105  EM130:  .ASCIZ  /UNEXPECTED TRAP TO 4/
21258 134460     105     122     122  EM131:  .ASCIZ  /ERROR WRITING TO LKS<6>/
21259 134510     105     122     122  EM132:  .ASCIZ  /ERROR IN MAINTENANCE REGISTER/
21260 134546     105     105     122  EM133:  .ASCIZ  /EEROM TYPE ERROR/
21261 134567     122     105     101  EM134:  .ASCIZ  /READ OR WRITE EEROM ERROR/
21262
21263 134621     040     124     105  DH1:    .ASCII  / TEST  ERROR/<15><12>
21264 134636     040     040     043          .ASCIZ  /  #     PC/
21265 134646     040     124     105  DH4:    .ASCII  / TEST  ERROR   EXPCTED RECEIVED/<15><12>
21266 134704     040     040     043          .ASCIZ  /  #     PC      DATA     DATA/
21267 134733     040     124     105  DH5:    .ASCII  / TEST  ERROR   HITMIS  DATA IN DATA IN/<15><12>
21268 134777     040     040     043          .ASCIZ  /  #     PC      REG.    CACHE   MEMORY/
21269 135032     040     124     105  DH7:    .ASCII  / TEST  ERROR   ADDRESS MSER/<15><12>
21270 135064     040     040     043          .ASCIZ  /  #     PC      ACCESSED/
21271 135105     040     124     105  DH24:   .ASCII  / TEST  ERROR   NUMBER/<15><12>
21272 135131     040     040     043          .ASCIZ  /  #     PC      OF HITS/
21273 135151     105     122     122  DH27:   .ASCII  \ERROR  ERROR   MSER    HIT/MISS\<15><12>
21274 135204     040     040     043          .ASCIZ  /  #     PC/
21275 135214     040     124     105  DH41:   .ASCII  / TEST  ERROR   INSTRUCTION/<15><12>
21276 135245     040     040     043          .ASCIZ  /  #     PC      OPCODE/
21277 135265     040     124     105  DH43:   .ASCII  / TEST  ERROR   EXPECTD RECEIVD CACHE/<15><12>
21278 135330     040     040     043          .ASCIZ  /  #     PC      DATA    DATA   LOCATION/
21279 135365     040     124     105  DH47:   .ASCII  / TEST  ERROR   ADDRESS ADDRESS/<15><12>
21280 135422     040     040     043          .ASCIZ  /  #     PC      <21-16>  <15-0>/
21281 135452     040     124     105  DH65:   .ASCII  / TEST  ERROR   PRIORITY/<15><12>
21282 135500     040     040     043          .ASCIZ  /  #     PC      LEVEL/
```

GLOBAL ERROR MESSAGES

```
21283 135516  040  124  105  DH72:   .ASCII  / TEST  ERROR   ADDRESS/<15><12>
21284 135543  040  040  043          .ASCIZ  /  #     PC     FAILED/
21285 135562  040  124  105  DH105:  .ASCII  / TEST  ERROR   RBUF/<15><12>
21286 135604  040  040  043          .ASCIZ  /  #     PC/
21287 135614  040  124  105  DH115:  .ASCII  / TEST  ERROR   MSER    ADDRESS/<15><12>
21288 135646  040  040  043          .ASCIZ  /  #     PC             ACCESSED/
21289 135671  040  124  105  DH134:  .ASCII  / TEST  ERROR    # of                  DATA  PCR   ADDRESS/<15><12>
21290 135744  040  040  043          .ASCIZ  /  #     PC     ERRORS  PATTERN    READ        ACCESSED/
21291
21292                                 .EVEN
21293 136026  001162  001116  000000  DT1:    .WORD   $TMP1,$ERRPC,0
21294 136034  001162  001116  000001  DT4:    .WORD   $TMP1,$ERRPC,R1,CCR,0
21295 136046  001162  001116  000002  DT5:    .WORD   $TMP1,$ERRPC,R2,R1,$GDDAT,0
21296 136062  001162  001116  001122  DT7:    .WORD   $TMP1,$ERRPC,$BDADR,MSER,0
21297 136074  001162  001116  001124  DT14:   .WORD   $TMP1,$ERRPC,$GDDAT,TSTLOC,0
21298 136106  001162  001116  001124  DT17:   .WORD   $TMP1,$ERRPC,$GDDAT,RECDAT,0
21299 136120  001162  001116  000003  DT24:   .WORD   $TMP1,$ERRPC,R3,0
21300 136130  001162  001116  177744  DT27:   .WORD   $TMP1,$ERRPC,MSER,R3,0
21301 136142  001162  001116  001124  DT35:   .WORD   $TMP1,$ERRPC,$GDDAT,MSER,0
21302 136154  001162  001116  001126  DT41:   .WORD   $TMP1,$ERRPC,$BDDAT,0
21303 136164  001162  001116  000001  DT43:   .WORD   $TMP1,$ERRPC,R1,RECDAT,$BDADR,0
21304 136200  001162  001116  172354  DT47:   .WORD   $TMP1,$ERRPC,KIPAR6,$BDADR,0
21305 136212  001162  001116  000001  DT50:   .WORD   $TMP1,$ERRPC,R1,$BDADR,0
21306 136224  001162  001116  001124  DT51:   .WORD   $TMP1,$ERRPC,$GDDAT,PCR,0
21307 136236  001162  001116  001124  DT52:   .WORD   $TMP1,$ERRPC,$GDDAT,BCSR,0
21308 136250  001162  001116  001124  DT64:   .WORD   $TMP1,$ERRPC,$GDDAT,LKSFL,0
21309 136262  001162  001116  001124  DT65:   .WORD   $TMP1,$ERRPC,$GDDAT,0
21310 136272  001162  001116  001124  DT75:   .WORD   $TMP1,$ERRPC,$GDDAT,$BDDAT,0
21311 136304  001162  001116  177562  DT105:  .WORD   $TMP1,$ERRPC,RBUF,0
21312 136314  001162  001116  001126  DT115:  .WORD   $TMP1,$ERRPC,$BDDAT,KIPAR6,$BDADR,0
21313 136330  001162  001122  000000  DT130:  .WORD   $TMP1,$BDADR,0
21314 136336  001162  001116  003022  DT134:  .WORD   $TMP1,$ERRPC,ERRCNT,R3,$BDDAT,R4,$BDADR,0
```

```
21316                                   .SBTTL  MODIFIED ERROR MESSAGE TYPEOUT ROUTINE
21317                                   ;;**************************************************************
21318                                   ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
21319                                   ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
21320                                   ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
21321                                   ;*
21322                                   ;*THE ONLY DIFFERENCE BETWEEN THIS ROUTINE AND THE ORIGINAL "$ERRTYP" FROM
21323                                   ;*SYSMAC IS THAT YOU CAN PASS INFORMATION IN GENERAL PURPOSE REGISTERS TO THIS
21324                                   ;*ROUTINE. THE GENERAL PURPOSE REGISTERS USED ARE TO BE SPECIFIED IN DT*
21325                                   ;*FORMAT. R0 SHOULD NOT BE USED.
21326
21327 136356                           ERTYPE:
21328 136356  005037  001162                    CLR     $TMP1           ;;JUST CLEAR IT
21329 136362  113737  001102  001162            MOVB    $TSTNM,$TMP1    ;;STORE TEST NUMBER
21330 136370  104401  001175                    TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
21331 136374  010046                          MOV     R0,-(SP)        ;;SAVE R0
21332 136376  005000                          CLR     R0              ;;PICKUP THE ITEM INDEX
21333 136400  153700  001114                BISB    @#$ITEMB,R0
21334 136404  001004                          BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
21335                                                                 ;;TYPE THE PC OF THE ERROR
21336 136406  013746  001116                MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
21337                                                                 ;;ERROR ADDRESS
21338 136412  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
21339 136414  000426                          BR      6$              ;;GET OUT
21340 136416  005300                  1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
21341 136420  006300                          ASL     R0              ;;      WORK FOR THE ERROR TABLE
21342 136422  006300                          ASL     R0
21343 136424  006300                          ASL     R0
21344 136426  062700  001324                ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
21345 136432  012037  136442                MOV     (R0)+,2$        ;;PICKUP "ERROR MESSAGE" POINTER
21346 136436  001404                          BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
21347 136440  104401                          TYPE
21348 136442  000000                  2$:     .WORD   0               ;;ERROR MESSAGE POINTER GOES HERE
21349 136444  104401  001175                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
21350 136450  012037  136460          3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
21351 136454  001404                          BEQ     5$              ;;SKIP TYPEOUT IF 0
21352 136456  104401                          TYPE                    ;;TYPE THE "DATA HEADER"
21353 136460  000000                  4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
21354 136462  104401  001175                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
21355 136466  011000                  5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
21356 136470  001004                          BNE     7$              ;;GO TYPE THE DATA
21357 136472  012600                  6$:     MOV     (SP)+,R0        ;;RESTORE R0
21358 136474  104401  001175                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
21359 136500  000207                          RTS     PC              ;;RETURN
21360 136502                          7$:
21361 136502  021027  000005                CMP     (R0),#5         ;;GENERAL PURPOSE REGISTER?
21362 136506  101021                          BHI     9$              ;;IF NOT, GO TYPE DATA
21363 136510  042737  000700  136544          BIC     #BIT8!BIT7!BIT6,8$      ;;CLEAR BITS FOR SOURCE REGISTER
21364 136516  011037  001160                MOV     (R0),$TMP0      ;;SAVE (R0)
21365 136522  000337  001160                SWAB    $TMP0           ;;GET REGISTER NUMBER TO HIGH BYTE
21366 136526  006237  001160                ASR     $TMP0           ;;GET REGISTER NUMBER TO BITS 8-6
21367 136532  006237  001160                ASR     $TMP0
21368 136536  053737  001160  136544          BIS     $TMP0,8$        ;;SET BITS IN MOV INSTRUCTION
21369                                                                 ;;ACCORDING TO REGISTER NUMBER
21370 136544  010046                  8$:     MOV     R0,-(SP)        ;;MOVE CONTEXT OF REGISTER TO STACK
21371 136546  005720                          TST     (R0)+           ;;ADVANCE POINTER
21372 136550  000401                          BR      10$             ;;GO TYPE
```

MODIFIED ERROR MESSAGE TYPEOUT ROUTINE

```
21373 136552  013046                  9$:     MOV     @(R0)+,-(SP)    ;;IF NOT GPR, SAVE @(R0)+ FOR TYPEOUT
21374 136554  104402                  10$:    TYPOC                   ;;GO TYPE--OCTAL ASCI (ALL DIGITS)
21375 136556  005710                          TST     (R0)            ;;IS THERE ANOTHER NUMBER?
21376 136560  001744                          BEQ     6$              ;;BR IF NO
21377 136562  104401  136570                  TYPE    ,11$            ;;TYPE TWO(2) SPACES
21378 136566  000745                          BR      7$
21379 136570     040     040     000  11$:    .ASCIZ  /  /            ;;TWO(2) SPACES
21380                                          .EVEN
21381
21382                                 .SBTTL  GLOBAL SUBROUTINES SECTION
21383
21384                                 ;++
21385                                 ; THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
21386                                 ; THAT ARE USED IN MORE THAN ONE TEST.
21387                                 ;--
21388
21389                                 ;++
21390                                 ; FUNCTIONAL DESCRIPTION:
21391                                 ;    SUBROUTINE TO INITIALIZE ALL THE MMU REGISTERS
21392
21393
21394                                 ; INPUTS: NONE
21395
21396                                 ; OUTPUTS: NONE
21397
21398                                 ; SUBORDINATE ROUTINES USED: LOAD PARS
21399                                 ;                            LOAD PDRS
21400
21401                                 ; FUNCTIONAL SIDE EFFECTS: NONE
21402
21403                                 ; CALLING SEQUENCE:     JSR     PC,INITMM
21404
21405 136574  012701  172240    INITMM: MOV     #172240,R1              ;BASE ADDRESS OF SIPARS
21406 136600  004737  136736            JSR     PC,     LDPARS
21407 136604  012701  172260            MOV     #172260,R1              ;BASE ADDRESS OF SDPARS
21408 136610  004737  136736            JSR     PC,     LDPARS
21409 136614  012701  172340            MOV     #172340,R1              ;BASE ADDRESS OF KIPARS
21410 136620  004737  136736            JSR     PC,     LDPARS
21411 136624  012701  172360            MOV     #172360,R1              ;BASE ADDRESS OF KDPARS
21412 136630  004737  136736            JSR     PC,     LDPARS
21413 136634  012701  177640            MOV     #177640,R1              ;BASE ADDRESS OF UIPARS
21414 136640  004737  136736            JSR     PC,     LDPARS
21415 136644  012701  177660            MOV     #177660,R1              ;BASE ADDRESS OF UDPARS
21416 136650  004737  136736            JSR     PC,     LDPARS
21417 136654  012701  177600            MOV     #177600,R1              ;BASE ADDRESS OF UIPDRS
21418 136660  004737  136766            JSR     PC,     LDPDRS
21419 136664  012701  177620            MOV     #177620,R1              ;BASE ADDRESS OF UDPDRS
21420 136670  004737  136766            JSR     PC,     LDPDRS
21421 136674  012701  172300            MOV     #172300,R1              ;BASE ADDRESS OF KIPDRS
21422 136700  004737  136766            JSR     PC,     LDPDRS
21423 136704  012701  172320            MOV     #172320,R1              ;BASE ADDRESS OF KDPDRS
21424 136710  004737  136766            JSR     PC,     LDPDRS
21425 136714  012701  172200            MOV     #172200,R1              ;BASE ADDRESS OF SIPDRS
21426 136720  004737  136766            JSR     PC,     LDPDRS
21427 136724  012701  172220            MOV     #172220,R1              ;BASE ADDRESS OF SDPDRS
21428 136730  004737  136766            JSR     PC,     LDPDRS
21429 136734  000207                    RTS     PC                      ;RETURN
```

GLOBAL SUBROUTINES SECTION

```
21431                                   ;**
21432                                   ; FUNCTIONAL DESCRIPTION:
21433                                   ;    SUBROUTINE TO INITIALIZE ALL THE MMU PAGE ADDRESS REGISTERS (PARS).
21434                                   ;       THIS ROUTINE WILL INITIALIZE 8 PARS STARTING AT A BASE ADDRESS
21435                                   ;       SUPPLIED BY THE CALLING ROUTINE. PARS 0-5 WILL BE MAPPED FROM
21436                                   ;       ADDRESS 0 TO ADDRESS 137777 (0-24K). PAR 6 WILL BE MAPPED FROM
21437                                   ;       ADDRESS 200000 TO 217777 AND PAR 7 WILL BE MAPPED TO THE I/O
21438                                   ;       PAGE.
21439                                   ;
21440                                   ; INPUTS:
21441                                   ;       R1 CONTAINS THE BASE ADDRESS OF THE NEXT 8 PARS TO BE INITIALIZED
21442                                   ;
21443                                   ; OUTPUTS: NONE
21444                                   ;
21445                                   ; SUBORDINATE ROUTINES USED: NONE
21446                                   ;
21447                                   ; FUNCTIONAL SIDE EFFECTS: NONE
21448                                   ;
21449                                   ; CALLING SEQUENCE:     JSR     PC,LDPARS
21450                                   ;
21451 136736  012702  000006    LDPARS: MOV     #6,     R2              ;LET LOOP COUNTER COUNT FIRST 6 PARS
21452 136742  005003                    CLR     R3                      ;INITIALIZE INDEX VALUE
21453 136744  010321            1$:     MOV     R3,     (R1)+           ;LOAD PARS
21454 136746  062703  000200            ADD     #200,   R3              ;INDEX IN 4K INCREMENTS
21455 136752  077204                    SOB     R2,     1$              ;LOAD FIRST SIX PARS
21456 136754  012721  002000            MOV     #2000,  (R1)+           ;LET PAR6 MAP TO 200000
21457 136760  012711  177600            MOV     #177600,(R1)            ;LET PAR7 MAP TO I/O PAGE
21458 136764  000207                    RTS     PC                      ;RETURN
```

```
                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;    SUBROUTINE TO INITIALIZE ALL THE MMU PAGE DECRIPTOR REGISTERS (PDRS).
                                ;       THIS ROUTINE WILL INITIALIZE 8 PDRS STARTING AT A BASE ADDRESS
                                ;       SUPPLIED BY THE CALLING ROUTINE. PDRS 0-5 WILL BE INITIALIZED TO
                                ;       4K READ/WRITE BYPASS AND PDRS 6 AND 7 WILL BE INITIALIZED TO
                                ;       4K READ/WRITE NO BYPASS.
                                ;       NOTE:   THERE IS NO NEED TO BYPASS ON I/O PAGE REFERENCES BECAUSE
                                ;               THE CACHE DOES NOT ALLOCATE ANY OF THESE REFERENCES.
                                ;
                                ; INPUTS:
                                ;       R1 CONTAINS THE BASE ADDRESS OF THE NEXT 8 PDRS TO BE INITIALIZED
                                ;
                                ; OUTPUTS: NONE
                                ;
                                ; SUBORDINATE ROUTINES USED: NONE
                                ;
                                ; FUNCTIONAL SIDE EFFECTS: NONE
                                ;
                                ; CALLING SEQUENCE:     JSR     PC,LDPARS
                                ;
136766  012702  000006        LDPDRS: MOV     #6,     R2              ;LET LOOP COUNTER COUNT FIRST 6 PARS
136772  012721  177406        1$:     MOV     #177406,(R1)+           ;LOAD PDRS WITH 4K READ/WRITE BYPASS
136776  077203                        SOB     R2,     1$              ;LOAD FIRST SIX PDRS
137000  012721  077406                MOV     #77406, (R1)+           ;LET PAR6 BE 4K READ/WRITE NO BYPASS
137004  012711  077406                MOV     #77406, (R1)            ;LET PAR7 BE 4K READ/WRITE NO BYPASS ALSO
137010  000207                        RTS     PC                      ;RETURN
```

GLOBAL SUBROUTINES SECTION

```
21488                                   ;++
21489                                   ; FUNCTIONAL DESCRIPTION:
21490                                   ;    SUBROUTINE TO HANDLE PARITY ERROR ABORTS FROM THE RAM STORE RAM TESTS.
21491
21492                                   ; INPUTS:
21493                                   ;       MEMORY SYSTEM ERROR REGISTER CONTAINS BITS INDICATING FAILURE
21494
21495                                   ; OUTPUTS: NONE
21496
21497                                   ; SUBORDINATE ROUTINES USED: NONE
21498
21499                                   ; FUNCTIONAL SIDE EFFECTS: NONE
21500
21501                                   ; CALLING SEQUENCE: CALLED BY PARITY ABORT
21502                                   ;       MOV     @#114,  SLOCOO  ;SAVE CONTENTS OF PARITY ABORT VECTOR
21503                                   ;       MOV     #DSPAR, @#114   ;LET VECTOR POINT TO PARITY ABORT ROUTINE
21504                                   ;
21505                                   ;       (CACHE PARITY ERROR OCCURS)
21506
21507 137012  011637  001122          RAMPAR: MOV     (SP),$BDADR             ;STOR ADDESS TRAPPED
21508 137016  032737  000100  177744          BIT     #BIT06, MSER            ;IF LOW BYTE PARITY ERROR
21509 137024  001401                        BEQ     1$                      ;THEN
21510 137026  104007                        ERROR   .7                      ;ERROR
21511 137030  032737  000200  177744  1$:     BIT     #BIT07, MSER            ;IF HIGH BYTE PARITY ERROR
21512 137036  001401                        BEQ     2$                      ;THEN
21513 137040  104007                        ERROR   .7                      ;ERROR
21514 137042  032737  000040  177744  2$:     BIT     #BIT05, MSER            ;IF TAG PARITY ERROR
21515 137050  001401                        BEQ     3$                      ;THEN
21516 137052  104007                        ERROR   .7                      ;ERROR
21517 137054  005037  177744          3$:     CLR     MSER                    ;INITIALIZE MSER AFTER ERROR
21518 137060  000002                        RTI                             ;RETURN
21519 137062  005237  002722          LKSINT: INC     LKSFL                   ;INCREMENT FLAG
21520 137066  000002                        RTI
21521
21522                                   .SBTTL Q22BE SIZE ROUTINE
21523                                   ;THIS ROUTINE WILL AUTOSIZE FOR UP TO TWO Q22 BUS EXERCISERS.  IF NONE
21524                                   ;FOUND LOCATIONS CSR1 AND CSR12 WILL BE LEFT ZEROES.  THIS ROUTINE WILL
21525                                   ;ONLY RUN IN NOT UFD MODE.
21526
21527 137070  032737  001000  177750  Q22SIZ: BIT     #BIT09,MAIREG                   ;UNIBUS SYSTEM?
21528 137076  001401                        BEQ     1$                              ;IF NOT, ADVANCE TO ROUTINE
21529 137100  000207                        RTS     PC                              ;OTHERWISE, RETURN
21530                                   ;
21531                                   ; PREPARE TO DO SIZING
21532                                   ;
21533 137102  013701  000004          1$:     MOV     ERRVEC,R1                       ;STORE TIMEOUT VECTOR
21534 137106  012737  137262  000004          MOV     #7$,ERRVEC                      ;POINT NEW TO PROGRAM
21535 137114  012737  000340  000006          MOV     #340,ERRVEC+2                   ;AT PRIORITY 7
21536 137122  005037  001160                  CLR     $TMP0                           ;CLEAR Q22BE COUNTER
21537 137126  012702  170000                  MOV     #170000,R2                      ;FIRST POSSIBLE ADDRESS
21538 137132  012703  000510                  MOV     #510,R3                         ;VECTOR FOR IT
21539 137136  000404                        BR      3$                              ;TRY THOSE VALUES
21540                                   ;
21541                                   ; NOW DO ACTUAL SIZING
21542                                   ;
21543 137140  062702  000020          2$:     ADD     #20,R2                          ;GET CSR FOR NEXT Q22BE
21544 137144  062703  000004                  ADD     #4,R3                           ;GET VECTOR FOR NEXT ONE
```

Q22BE SIZE ROUTINE

```
21545 137150  005712                    3$:     TST     (R2)                    ;TRY TO ACCESS CSR
21546                                   ;
21547                                   ; IF NO TIMEOUT, STORE EXISTING ADDRESSES TO REGISTERS
21548                                   ;
21549 137152  005737  001160                    TST     $TMPO                   ;FIRST Q22BE FOUND?
21550 137156  001010                            BNE     4$                      ;IF SECOND, BRANCH
21551 137160  012705  002664                    MOV     #CSR1,R5                ;START WITH CSR1 FOR 1ST
21552 137164  010237  002702                    MOV     R2,SIMGOA               ;SIMULTANEOUS GO
21553 137170  062737  000016  002702            ADD     #16,SIMGOA              ;ADDRESS
21554 137176  000402                            BR      5$                      ;BRANCH TO INITIALISE
21555 137200  012705  002704            4$:     MOV     #CSR12,R5               ;START WITH CSR12 FOR 2ND
21556 137204  012704  000004            5$:     MOV     #4,R4                   ;INITIALISE 5 REGISTERS
21557 137210  010215                            MOV     R2,(R5)                 ;INITIALISE CSR1
21558 137212  011565  000002            6$:     MOV     (R5),2(R5)              ;STORE TO NEXT ONE
21559 137216  005725                            TST     (R5)+                   ;GET NEXT ADDRESS
21560 137220  062715  000002                    ADD     #2,(R5)                 ;GET ADDRESS, POINT NEXT
21561 137224  077406                            SOB     R4,6$                   ;DO FOR NEXT 4 REGISTERS
21562 137226  010365  000002                    MOV     R3,2(R5)                ;STORE INTERRUPT VECTOR
21563 137232  010365  000004                    MOV     R3,4(R5)                ;AND PRIORITY
21564 137236  062765  000002  000004            ADD     #2,4(R5)
21565 137244  005237  001160                    INC     $TMPO                   ;COUNT Q22BE'S
21566 137250  022737  000002  001160            CMP     #2,$TMPO                ;TWO FOUND?
21567 137256  001406                            BEQ     9$                      ;IF SO, STOP SIZING
21568 137260  000402                            BR      8$                      ;OTHERWISE, CONTINUE SIZING
21569                                   ;
21570                                   ; ON TIMEOUT TRY TO LOOK AT NEXT ADDRESS RANGE
21571                                   ;
21572 137262  005726                    7$:     TST     (SP)+                   ;RESTORE STACK FROM
21573 137264  005726                            TST     (SP)+                   ;TIMEOUT
21574 137266  022702  170160            8$:     CMP     #170160,R2              ;AT THE LAST POSSIBLE?
21575 137272  001322                            BNE     2$                      ;IF NOT, BRANCH
21576 137274  005737  002664            9$:     TST     CSR1                    ;1 FOUND?
21577 137300  001402                            BEQ     10$                     ;IF NONE, BRANCH
21578 137302  104401  137314                    TYPE    ,ONOQ22                 ;TYPE FOUND
21579 137306  010137  000004            10$:    MOV     R1,ERRVEC               ;RESTORE TIMEOUT VECTOR
21580 137312  000207                            RTS     PC                      ;RETURN
21581
21582 137314      012      015      121  ONOQ22: .ASCIZ  <12><15>/Q22BE USED DURING TESTING/
21583                                   .EVEN
21584                                   .SBTTL Q22BE INTERRUPT INITIALISE ROUTINE
21585                                   ;THIS ROUTINE WILL INITIALISE Q22BE TO INTERRUPT AT A PRIORITY AT (R0)+
21586                                   ;AT THE STARTING ADDRESS IN R3. THE TEST HAVE TO SET ACTUAL DONE BIT
21587                                   ;BY CLEARING GO.
21588
21589 137350  005013                    Q22INT: CLR     (R3)                    ;CLEAR TRANSFER TYPE IN CSR1
21590 137352  052710  000001                    BIS     #BIT00,(R0)             ;ZERO DONE
21591 137356  011063  000002                    MOV     (R0),2(R3)              ;SET PRIORITY IN CSR2
21592 137362  042710  000001                    BIC     #BIT00,(R0)             ;PREPARE TO SET DONE
21593 137366  000207                            RTS     PC
21594
21595                                   .SBTTL DMATRN DATO CYCLE THRU Q22BE
21596                                   ;THIS ROUTINE PERFORMS DATO FROM A LOCATION TEMP THRU THE FIRST
21597                                   ;FOUND Q22BE STARTING AT LOCATION @CSR1. R0 HAS 0 IF ONLY 1 TRANSFER IS
21598                                   ;TO BE PERFORMED. OTHERWISE 16 BLOCK MODE TRANSFERS ARE TO BE PERFORMED.
21599                                   ;IN THE LATTER CASE ADDRESS AND WORD COUNT HAS TO BE LOADED BEFORE.
21600
21601 137370  012777  012525  043276  DMATRN: MOV     #12525,@DATA            ;DATA USED
```

DMATRN DATO CYCLE THRU Q22BE

```
21602 137376  005700                        TST     R0              ;DO 1 WORD?
21603 137400  001404                        BEQ     1$              ;IF YES, BRANCH
21604 137402  012777  001001  043256        MOV     #BIT09!BIT00,@CSR2      ;BLOCK MODE, GO
21605 137410  000414                        BR      2$              ;BRANCH TO DO IT
21606 137412  012777  001601  043244  1$:   MOV     #1601,@CSR1     ;RESET LATENCY COUNT,DATO
21607 137420  012777  002740  043242        MOV     #TEMP,@BA       ;LOAD DMA ADDRESS
21608 137426  012777  177777  043236        MOV     #177777,@WC     ;DO 1 WORD
21609 137434  012777  000001  043224        MOV     #BIT00,@CSR2    ;DO IT
21610 137442  105777  043220          2$:   TSTB    @CSR2           ;DMA DONE?
21611 137446  100375                        BPL     2$              ;WAIT TILL DONE
21612 137450  000207                  3$:   RTS     PC              ;RETURN FROM SUBROUTINE
21613
21614                                 .SBTTL DMARD DATI THRU Q22BE
21615                                 ;THIS ROUTINE PERFORMS DATI CYCLE THRU Q22BE IN EITHER BLOCK MODE OR A SINGLE
21616                                 ;TRANSFER MODE. MEMORY LOCATION USED IS TEMP. R0 IS ZERO FOR SINGLE TRANSFER
21617
21618 137452  012777  002740  043210  DMARD: MOV     #TEMP,@BA       ;LOAD DMA ADDRESS
21619 137460  005700                        TST     R0              ;DO 1 WORD?
21620 137462  001412                        BEQ     1$              ;IF YES, BRANCH
21621 137464  012777  001507  043172        MOV     #1507,@CSR1     ;16 DATIB
21622 137472  012777  177770  043172        MOV     #177770,@WC     ;DO 8 WORD
21623 137500  012777  001001  043160        MOV     #BIT09!BIT00,@CSR2      ;BLOCK MODE GO
21624 137506  000411                        BR      2$              ;GO CHECK
21625 137510  012777  001407  043146  1$:   MOV     #1407,@CSR1     ;RESET LATENCY COUNT,DATI
21626                                                               ;LOAD NEW DATA TO DATA R.
21627 137516  012777  177777  043146        MOV     #177777,@WC     ;DO 1 WORD
21628 137524  012777  000001  043134        MOV     #BIT00,@CSR2    ;DO IT
21629 137532  105777  043130          2$:   TSTB    @CSR2           ;DMA DONE?
21630 137536  100375                        BPL     2$              ;WAIT TILL DONE
21631 137540  000207                  3$:   RTS     PC              ;RETURN FROM SUBROUTINE
21632
21633
21634
21635 137542  011637  001122  TOUT:   MOV     (SP),$BDADR     ;STORE TRAPPED PC
21636 137546  104130                  ERROR   +130            ;UNEXPECTED TRAP
21637 137550  000002                  RTI
21638
21640                                 ;
21641                                 ;MMU GLOBAL SUBROUTINES
21642                                 ;
21643                                 ;
21644                                 ;ROUTINE TO INITIALIZE MEMORY MANAGEMENT
21645                                 ;
21646 137552  010046          MMU:    MOV     R0,-(SP)        ;SAVE CONTENTS OF REGISTERS
21647 137554  010146                  MOV     R1,-(SP)        ;
21648 137556  010246                  MOV     R2,-(SP)        ;
21649 137560  012700  177600          MOV     #177600,R0      ;
21650 137564  004737  137652          JSR     PC,PDR          ;INIT I AND D USER PDR'S
21651 137570  004737  137674          JSR     PC,PAR          ;INIT I USER PAR'S
21652 137574  004737  137674          JSR     PC,PAR          ;INIT D USER PAR'S
21653 137600  012700  172200          MOV     #172200,R0
21654 137604  004737  137652          JSR     PC,PDR          ;INIT I AND D SUP PDR'S
21655 137610  004737  137674          JSR     PC,PAR          ;INIT I SUP PAR'S
21656 137614  004737  137674          JSR     PC,PAR          ;INIT D SUP PAR'S
21657 137620  004737  137652          JSR     PC,PDR          ;INIT I AND D KER PDR'S
21658 137624  004737  137674          JSR     PC,PAR          ;INIT I KER PAR'S
21659 137630  004737  137674          JSR     PC,PAR          ;INIT D KER PAR'S
```

DMARD DATI THRU Q22BE

```
21660 137634  012737  000027  172516          MOV     #27,@#172516            ;INIT MMR3
21661 137642  012602                          MOV     (SP)+,R2                ;RESTORE REGISTERS
21662 137644  012601                          MOV     (SP)+,R1                ;
21663 137646  012600                          MOV     (SP)+,R0                ;
21664 137650  000207                          RTS     PC                      ;RETURN
21665                                 ;
21666                                 ;ROUTINE TO INITIALIZE PDR'S
21667                                 ;
21668 137652  005002                  PDR:    CLR     R2                      ;INIT CNTR
21669 137654  012720  077406          PDR1:   MOV     #77406,(R0)+            ;INIT PDR
21670 137660  062702  000001                  ADD     #1,R2                   ;INCREMENT CNTR
21671 137664  022702  000020                  CMP     #16.,R2                 ;ARE WE DONE?
21672 137670  001371                          BNE     PDR1                    ;BRANCH IF NOT
21673 137672  000207                          RTS     PC                      ;RETURN
21674                                 ;
21675                                 ;ROUTINE TO INITIALIZE PAR'S
21676                                 ;
21677 137674  005001                  PAR:    CLR     R1                      ;SETUP TO INIT PAR
21678 137676  010120                  PAR1:   MOV     R1,(R0)+                ;INIT PAR
21679 137700  062701  000200                  ADD     #200,R1                 ;GET READY FOR NEXT PAR
21680 137704  022701  001600                  CMP     #1600,R1                ;REACHED A PAR7
21681 137710  001372                          BNE     PAR1                    ;BRANCH IF NOT
21682 137712  012720  177600                  MOV     #177600,(R0)+           ;INIT PAR7
21683 137716  000207                          RTS     PC                      ;RETURN
21684                                 ;
21685                                 ;TIME OUT ROUTINE
21686                                 ;
21687 137720  005205                  ADDTRP: INC     R5                      ;INCREMENT TIME OUT FLAG
21688 137722  000002                          RTI                             ;RETURN
21689                                 ;
21690                                 ;MMU TRAP ROUTINE
21691                                 ;
21692 137724  023727  003030  000001  MMUTRP: CMP     FLAG,#1                 ;ARE WE EXPECTING AN ABORT
21693 137732  001401                          BEQ     1$                      ;YES GO ON
21694 137734  104002                          ERROR   +2              ;NO GO TO ERROR
21695 137736  010046                  1$:     MOV     R0,-(SP)                ;SAVE CONTENTS OF REG 0
21696 137740  013700  177776                  MOV     @#177776,R0             ;SAVE A COPY OF PSW
21697 137744  072027  177764                  ASH     #-14,R0                 ;LOOK AT BITS<15:14>
21698 137750  020027  000002                  CMP     R0,#2                   ;WAS PS<15:14>=10
21699 137754  001001                          BNE     OK                      ;NO GO ON
21700 137756  000411                          BR      NOTOK                   ;YES CHANGE BITS TO 00
21701 137760  013700  177776          OK:     MOV     @#177776,R0             ;SAVE A COPY OF PSW
21702 137764  072027  000002                  ASH     #2,R0                   ;LOOK AT BITS<13:12>
21703 137770  072027  177764                  ASH     #-14,R0                 ;
21704 137774  020027  000002                  CMP     R0,#2                   ;WAS PS<13:12>=10
21705 140000  001002                          BNE     OK1                     ;NO GO ON
21706 140002  005066  000004          NOTOK:  CLR     4(SP)                   ;CLEAR ILLEGAL MODE FFROM OLD PSW
21707 140006  013737  177572  003042  OK1:    MOV     @#177572,SAVMR0         ;SAVE A COPY OF MMR0
21708 140014  013737  177574  003044          MOV     @#177574,SAVMR1         ;SAVE A COPY OF MMR1
21709 140022  013737  177576  003046          MOV     @#177576,SAVMR2         ;SAVE A COPY OF MMR2
21710 140030  005037  177572                  CLR     @#177572                ;CLEAR ABORT BITS AND TURN MMU OFF
21711 140034  005037  003030                  CLR     FLAG                    ;CLEAR MMU ABORT FLAG
21712 140040  012600                          MOV     (SP)+,R0                ;RESTORE ORIGINAL CONTENTS OF REG 0
21713 140042  000002                          RTI                             ;RETURN
```

```
21717                                   ;FPP COMMON SUBROUTINES
21718 140044  012600                    WLDTRP: MOV     (SP)+,R0                ;SAVE PC
21719 140046  012605                            MOV     (SP)+,R5                ;SAVE STATUS AND RESTORE STACK
21720 140050  104003                            ERROR   +3
21721 140052  000110                            JMP     (R0)                    ;GO BACK INLINE
21722                                   ;
21723                                   ;
21724                                   ;
21725 140054  000000                    TRPFLG: .WORD   0
21726 140056  000207                    ERRFP:  RTS     R7
21727 140060  000207                    ERR:    RTS     R7
21728                                   ;
21729                                   ;
21730                                   ;
21731                                   ;
21732                                   ;
21733                                   ;SUBROUTINE DATA VERFICATION -
21734                                   ;
21735                                   ; CALLED BY     JSR     R7,DATVER
21736                                   ;
21737                                   ;INPUT:         (R4)=EXPECTED DATA
21738                                   ;               (R1)=RECEIVED DATA
21739                                   ;
21740                                   ;THIS ROUTINE VERIFIES THAT THE 4 CONSECTIVE WORDS STARTING WITH (R4) ARE
21741                                   ;EQUAL TO THE FOUR WORDS ADDRESSED BY (R1). THE CONTENTS OF R4, AND R1 ARE NOT
21742                                   ;DISTURBED.
21743                                   ;LOCATION "COUNT" , IF NOT EQUAL TO 0 SIGNIFIES DATA ERROR
21744                                   ;IF THE STATUS IS FLOATING MODE, THE LAST TWO BYTES OF RECEIEVED
21745                                   ;ARE SIMPLY CHECKED FOR ZEROS
21746                                   ;
21747
21748 140062  010446                    DATVFR: MOV     R4,-(SP)                ;SAVE R4
21749 140064  010146                            MOV     R1,-(SP)                        ;SAVE R1
21750 140066  012737  000003  003120            MOV     #3,COUNT                ;SET UP ITERATION COUNT
21751 140074  000137  140112                    JMP     DAT1                    ;
21752                                   ;
21753 140100  010446                    DATVER: MOV     R4,-(SP)                ;SAVE R4
21754 140102  010146                            MOV     R1,-(SP)                ;SAVE R1
21755 140104  012737  000005  003120            MOV     #5,COUNT                ;SET UP ITERATION COUNT
21756 140112  005337  003120            DAT1:   DEC     COUNT
21757 140116  001402                            BEQ     2$                      ;BRANCH IF DONE
21758 140120  022421                            CMP     (R4)+,(R1)+                     ;
21759 140122  001773                            BEQ     DAT1
21760 140124  012601                    2$:     MOV     (SP)+,R1                ;RESTORE R1
21761 140126  012604                            MOV     (SP)+,R4                ;RESTORE R4
21762 140130  000207                            RTS     R7                      ;GO BACK TO CALLING ROUTINE
21763                                                                           ;IF DATA ERROR, COUNT NE 0
```

```
21765                                   ;
21766                                   ;$$$
21767                                   ;
21768                                   ; SUBROUTINE - DETERMINE FLOATING POINT ACCELERATOR (DETFPA)
21769                                   ;
21770                                   ; THIS SUBROUTINE IS CALLED IF AN ERROR IS DETECTED DURING EXECUTION OF THE
21771                                   ; FLOATING POINT TESTS.
21772                                   ; IT DETERMINES WHEATHER OR NOT THE FLOATING POINT ACCELERATOR CHIP OPTION
21773                                   ; IS PRESENT ON THE CPU BOARD AND PRINTS THE APPROPRIATE ERROR MESSAGE.
21774                                   ; THIS DETERMINATION IS MADE BASED ON THE "FPA AVAILABLE" FLAG, BIT 8
21775                                   ; OF THE MAINTENANCE REGISTER AT LOCATION 17777750. IF THE FPA BIT IS SET
21776                                   ; THEN THE FLOATING POINT ACCELERATOR CHIP IS INSTALLED ON THE CPU BOARD AND
21777                                   ; AN ERROR MESSAGE IS PRINTED WHICH STATES THAT THE FLOATING POINT ERROR IS
21778                                   ; DUE TO THIS CHIP. OTHERWISE, THE J11 IS BLAMED FOR THE FLOATING POINT ERROR.
21779                                   ;
21780                                   ;$$$
21781                                   ;
21782                                   ; CALLED BY: CALL       @#DETFPA ;$$$
21783                                   ;
21784                                   ; INPUTS: NONE                   ;$$$
21785                                   ;
21786                                   ; OUTPUTS: ERROR MESSAGES        ;$$$
21787                                   ;
21788 140132  032737  000400  177750  DETFPA: BIT     #400,@#MAIREG   ;IS THE FPA HERE? $$$
21789 140140  001007                          BNE     FPAOPT          ;YES, BRANCH FPAOPT $$$
21790 140142  000240                          NOP                     ;DEBUG AID. $$$
21791 140144  032737  000400  177750          BIT     #400,@#MAIREG   ;IF NOT, $$$
21792 140152  001405                          BEQ     NOFPA           ;BRANCH TO NOFPA $$$
21793 140154  000240                          NOP                     ;DEBUG AID. $$$
21794 140156  000000                          HALT                    ; $$$
21795                                   ;
21796 140160  104401  140174          FPAOPT: TYPE    ,FPAFLT         ; $$$
21797 140164  000402                          BR      EXTFPA          ; $$$
21798 140166  104401  140257          NOFPA:  TYPE    ,J11FLT         ; $$$
21799 140172  000207                  EXTFPA: RTS     PC              ; $$$
21800                                   ;
21801 140174      105      122      122  FPAFLT: .ASCIZ  /ERROR DETECTED IN FLOATING POINT ACCELERATOR CHIP./ ; $$$
21802                                   ;
21803 140257      105      122      122  J11FLT: .ASCIZ  /ERROR DETECTED IN J11 FLOATING POINT PROCESSOR./ ; $$$
21804                                           .EVEN                   ; $$$
21805                                   ;
```

DMARD DATI THRU Q22BE

```
21808
21809
21810                                   .SBTTL  END OF PASS ROUTINE
                                        ;;*****************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
                                        ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO LOOP
140340                                  $EOP:
140340  032737  000100  000052                BIT #BIT06,@#52
140346  001030                                BNE $GET42
140350  004537  126756                        jsr r5,vireop
140354  005037  001102                        CLR     $TSTNM          ;;ZERO THE TEST NUMBER
140360  005037  001164                        CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
140364  005237  001206                        INC     $PASS           ;;INCREMENT THE PASS NUMBER
140370  042737  100000  001206                BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
140376  005327                                DEC     (PC)+           ;;LOOP?
140400  000001                          $EOPCT: .WORD   1
140402  003022                                BGT     $DOAGN          ;;YES
140404  012737                                MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
140406  000001                          $ENDCT: .WORD   1
140410  140400                                $EOPCT
140412  104401  140457                        TYPE    ,$ENDMG         ;;TYPE "END PASS #"
140416  013746  001206                        MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
140422  104405                                TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
140424  104401  140454                        TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
140430  013700  000042                $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
140434  001405                                BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
140436  000005                                RESET                   ;;CLEAR THE WORLD
140440  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
140442  000240                                NOP                     ;;SAVE ROOM
140444  000240                                NOP                     ;;FOR
140446  000240                                NOP                     ;;ACT11
140450                                  $DOAGN:
140450  000137                                JMP     @(PC)+          ;;RETURN
140452  004740                          $RTNAD: .WORD   LOOP
140454     377     377     000          $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
140457     015     012     105          $ENDMG: .ASCIZ  <15><12>/END PASS #/
21811                                   .SBTTL  SCOPE HANDLER ROUTINE
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                        ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW14=1        LOOP ON TEST
                                        ;*SW11=1        INHIBIT ITERATIONS
                                        ;*SW09=1        LOOP ON ERROR
                                        ;*SW08=1        LOOP ON TEST IN SWR<5:0>
                                        ;*CALL
                                        ;*      SCOPE           ;;SCOPE=IOT
140474                                  $SCOPE:
140474  104407                                CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
140476  052737  001000  177520                BIS #1000,BCSR ;ENABLE
140504  032777  040000  040426          1$:   BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
140512  001117                                BNE     $OVER           ;;YES IF SW14=1
                                        ;#####START OF CODE FOR THE XOR TESTER#####
```

```
140514  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
140516  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
140522  012737  140542  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
140530  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
140534  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
140540  000466                          BR      $SVLAD          ;;GO TO THE NEXT TEST
140542  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
140544  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
140550  000426                          BR      7$              ;;LOOP ON THE PRESENT TEST
140552                          6$:;#####END OF CODE FOR THE XOR TESTER#####
140552  032777  000400  040360          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
140560  001407                          BEQ     2$              ;;BR IF NO
140562  017746  040352                  MOV     @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
140566  042716  000300                  BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
140572  122637  001102                  CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
140576  001465                          BEQ     $OVER           ;;BR IF YES
140600  105737  001103          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
140604  001421                          BEQ     3$              ;;BR IF NO
140606  123737  001115  001103          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
140614  101015                          BHI     3$              ;;BR IF NO
140616  032777  001000  040314          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
140624  001404                          BEQ     4$              ;;BR IF NO
140626  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
140634  000446                          BR      $OVER
140636  105037  001103          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
140642  005037  001164                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
140646  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
140650  032777  004000  040262  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
140656  001011                          BNE     1$              ;;BR IF YES
140660  005737  001206                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
140664  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
140666  005237  001104                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
140672  023737  001164  001104          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
140700  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
140702  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
140710  013737  140774  001164          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
140716  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
140722  113737  001102  001204          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
140730  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
140734  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
140740  005037  001166                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
140744  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
140752  013777  001102  040162  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
140760  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
140764  042737  001000  177520          BIC #1000,BCSR ;DISABLE
140772  000002                          RTI
140774  000001                  $MXCNT: 1                       ;;MAX. NUMBER OF ITERATIONS
21812                           .SBTTL  ERROR HANDLER ROUTINE
                                ;;**************************************************************
                                ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                ;*AND GO TO ERTYPE ON ERROR
                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                ;*SW15=1         HALT ON ERROR
                                ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                ;*SW10=1         BELL ON ERROR
```

ERROR HANDLER ROUTINE

```
                                ;*SW09=1        LOOP ON ERROR
                                ;*CALL
                                ;*      ERROR   +N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
140776                          $ERROR:
140776  005737  004120                  TST     UQUIET          ;;TEST FOR USER-QUIET MODE
141002  001403                          BEQ     9$              ;;BRANCH IF FIELD-SERVICE MODE
141004  005000                          CLR     R0              ;;IN CASE R0 HAS A #3 IN IT (↑C)
141006  004737  141220                  JSR     PC,ABORT        ;;TEST FOR ABORT CONDITION
141012                          9$:
141012  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
141014  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
141022  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
141026  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
141030  013777  001102  040104          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
141036  032777  002000  040074          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
141044  001402                          BEQ     1$              ;;NO - SKIP
141046  104401  001170                  TYPE    ,$BELL          ;;RING BELL
141052  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
141056  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
141062  162737  000002  001116          SUB     #2,$ERRPC
141070  117737  040022  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
141076  032777  020000  040034          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
141104  001004                          BNE     20$             ;;SKIP TYPEOUTS
141106  004737  136356                  JSR     PC,ERTYPE       ;;GO TO USER ERROR ROUTINE
141112  104401  001175                  TYPE    ,$CRLF
141116                          20$:
141116  122737  000001  001220          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
141124  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
141126  113737  001114  141140          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
141134  004737  141376                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
141140     000          21$:    .BYTE   0
141141     000                  .BYTE   0
141142  000777          22$:    BR      22$             ;;APT ERROR LOOP
141144  005777  037770          2$:     TST     @SWR            ;;HALT ON ERROR
141150  100002                          BPL     3$              ;;SKIP IF CONTINUE
141152  000000                          HALT                    ;;HALT ON ERROR!
141154  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
141156  032777  001000  037754  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
141164  001402                          BEQ     4$              ;;BR IF NO
141166  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
141172  005737  001166          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
141176  001402                          BEQ     5$              ;;BR IF NONE
141200  013716  001166                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
141204                          5$:
141204  022737  140440  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
141212  001001                          BNE     6$              ;;BRANCH IF NO
141214  000000                          HALT                    ;;YES
141216                          6$:
141216  000002                          RTI                     ;;RETURN
                                .SBTTL  ABORT ROUTINE FOR LCP/ORION UFD MODE
141220  005737  004116          ABORT:  TST     UFDFLG          ;TEST FOR USER FRIENDLY MODE
141224  001454                          BEQ     NOABRT          ;IF NOT UFD THEN CONTINUE NORMAL OPERATION
141226  020027  000032                  CMP     R0,#32          ;IS IT A ↑Z ?
141232  001443                          BEQ     ABORTZ          ;JUST GO BACK TO CHAIN IF IT IS (NO ERROR)
141234  020027  000003                  CMP     R0,#3           ;IS IS A ↑C ?
141240  001404                          BEQ     ABORTC          ;BR TO LOAD ↑C ON XXDP+ STACK (NO ERROR)
141242  005737  004120                  TST     UQUIET          ;TEST FOR USER-QUIET MODE
```

ABORT ROUTINE FOR LCP/ORION UFD MODE

```
141246  001443                          BEQ     NOABRT          ;IF FIELD-SERVICE MODE, CONTINUE NORMAL OPERATION
                                                                ; BECAUSE FIELD-SERVICE MODE DOES NOT QUIT ON ERROR
141250  000422                          BR      ABORTE          ;SET DRSERR THEN LEAVE
141252  013737  004112  000030  ABORTC: MOV     SAV30,30        ;RESTORE EMT LOCATION (30)
141260  013737  004114  000032          MOV     SAV32,32        ;RESTORE EMT PRIORITY LOCATION (32)
141266  104043                          EMT     +43             ;GET XXDP STACK LOC. INTO R0 FROM MONITOR
141270  005720                  1$:     TST     (R0)+           ;FIND END OF STACK
141272  001376                          BNE     1$
141274  112760  000057  177777          MOVB    #'/,-1(R0)      ;LOAD SLASH OVER ZERO
141302  112720  000136                  MOVB    #'↑,(R0)+       ;LOAD UPARROW
141306  112720  000103                  MOVB    #'C,(R0)+       ;LOAD C
141312  105010                          CLRB    (R0)            ;MAKE NEW END TO STACK
141314  000412                          BR      ABORTZ          ;NOW LEAVE
141316  013737  004112  000030  ABORTE: MOV     SAV30,30        ;RESTORE EMT LOCATION (30)
141324  013737  004114  000032          MOV     SAV32,32        ;RESTORE EMT PRIORITY LOCATION (32)
141332  104042                          EMT     +42             ;GET DCA LOCATION INTO R0 FROM MONITOR
141334  012760  177777  000042          MOV     #-1,42(R0)      ;SET A -1 INTO LOCATION DRSERR IN MONITOR
141342  013700  000042          ABORTZ: MOV     @#42,R0         ;AND PUT THE MONITOR RETURN ADDRESS IN R0
141346  005037  000042                  CLR     @#42            ;CLEAR MONITOR RETURN FLAG
141352  000137  140440                  JMP     $ENDAD          ;RETURN TO MONITOR-DO NOT PUSH STACK HERE
141356  000207                  NOABRT: RTS     PC              ;IF NOTUFD RETURN TO MAINLINE
21813                                   .SBTTL  APT COMMUNICATIONS ROUTINE
                                        ;;**************************************************************
141360  112737  000001  141624  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
141366  112737  000001  141622  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
141374  000403                          BR      $ATYC
141376  112737  000001  141624  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
141404                          $ATYC:
141404  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
141406  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
141410  105737  141622                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
141414  001450                          BEQ     5$              ;;IF NOT:  BR
141416  122737  000001  001220          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
141424  001031                          BNE     3$              ;;IF NOT:  BR
141426  132737  000100  001221          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
141434  001425                          BEQ     3$              ;;IF NOT:  BR
141436  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
141442  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
141450  005737  001200          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
141454  001375                          BNE     1$              ;;IF NOT:  WAIT
141456  010037  001214                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
141462  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
141464  001376                          BNE     2$
141466  163700  001214                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
141472  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
141474  010037  001216                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
141500  012737  000004  001200          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
141506  000413                          BR      5$
141510  017637  000004  141534  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
141516  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
141524  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
141530  004737  141626                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
141534  000000                  4$:     .WORD   0
141536                          5$:
141536  105737  141624          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
141542  001416                          BEQ     12$             ;;IF NOT:  BR
141544  005737  001220                  TST     $ENV            ;;RUNNING UNDER APT?
```

APT COMMUNICATIONS ROUTINE

```
141550  001413                          BEQ     12$             ;;IF NOT:  BR
141552  005737  001200          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
141556  001375                          BNE     11$             ;;IF NOT:  WAIT
141560  017637  000004  001202          MOV     @4(SP),$FATAL   ;;GET ERROR #
141566  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
141574  005237  001200                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
141600  105037  141624          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
141604  105037  141623                  CLRB    $LFLG           ;;CLEAR LOG FLAG
141610  105037  141622                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
141614  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
141616  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
141620  000207                          RTS     PC              ;;RETURN
141622     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
141623     000          $LFLG:  .BYTE   0               ;;LOG FLAG
141624     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
                                .EVEN
        000200          APTSIZE=200
        000001          APTENV=001
        000100          APTSPOOL=100
        000040          APTCSUP=040
 21814                  .SBTTL  TYPE ROUTINE
                        ;;**************************************************************************
                        ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                        ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                        ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                        ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                        ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                        ;*
                        ;*CALL:
                        ;*1) USING A TRAP INSTRUCTION
                        ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                        ;*OR
                        ;*      TYPE
                        ;*      MESADR
                        ;*
141626  105737  001157  $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
141632  100002                  BPL     1$              ;;BR IF YES
141634  000000                  HALT                    ;;HALT HERE IF NO TERMINAL
141636  000430                  BR      3$              ;;LEAVE
141640  010046          1$:     MOV     R0,-(SP)        ;;SAVE R0
141642  017600  000002          MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
141646  122737  000001  001220  CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
141654  001011                  BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
141656  132737  000100  001221  BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
141664  001405                  BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
141666  010037  141676          MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
141672  004737  141366          JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
141676  000000          61$:    .WORD   0               ;;MESSAGE ADDRESS
141700  132737  000040  001221  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
141706  001003                  BNE     60$             ;;YES,SKIP TYPE OUT
141710  112046          2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
141712  001005                  BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
141714  005726                  TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
141716  012600          60$:    MOV     (SP)+,R0        ;;RESTORE R0
141720  062716  000002  3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
141724  000002                  RTI                     ;;RETURN
141726  122716  000011  4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
```

TYPE ROUTINE

```
141732  001430                          BEQ     8$
141734  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
141740  001006                          BNE     5$
141742  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
141744  104401                          TYPE                    ;;TYPE A CR AND LF
141746  001175                          $CRLF
141750  105037  142156                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
141754  000755                          BR      2$              ;;GET NEXT CHARACTER
141756  004737  142040          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
141762  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
141766  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
141770  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
141774  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
142000  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
142002  004737  142040                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
142006  105337  142156                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
142012  000770                          BR      7$              ;;LOOP
                        ;HORIZONTAL TAB PROCESSOR
142014  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
142020  004737  142040          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
142024  132737  000007  142156          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
142032  001372                          BNE     9$              ;;TAB STOP
142034  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
142036  000724                          BR      2$              ;;GET NEXT CHARACTER
142040                  $TYPEC:
142040  105777  037100                  TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?          ;MJD001
142044  100022                          BPL     10$             ;;BR IF NOT                     ;MJD001
142046  017746  037074                  MOV     @$TKB,-(SP)     ;;GET CHAR                      ;MJD001
142052  042716  177600                  BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS         ;MJD001
142056  122716  000023                  CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                 ;MJD001
142062  001012                          BNE     102$            ;;BR IF NOT                     ;MJD001
142064                          101$:                           ;MJD001
142064  105777  037054                  TSTB    @$TKS           ;;WAIT FOR CHAR                 ;MJD001
142070  100375                          BPL     101$                                            ;MJD001
142072  117716  037050                  MOVB    @$TKB,(SP)      ;;GET CHAR                      ;MJD001
142076  042716  177600                  BIC     #177600,(SP)    ;;STRIP IT                      ;MJD001
142102  122716  000021                  CMPB    #$XON,(SP)      ;;WAS IT XON?                   ;MJD001
142106  001366                          BNE     101$            ;;BR IF NOT                     ;MJD001
142110                          102$:                           ;MJD001
142110  005726                          TST     (SP)+           ;;FIX STACK                     ;MJD001
142112                          10$:                            ;MJD001
142112  105777  037032                  TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
142116  100375                          BPL     10$                                             ;MJD001
142120  116677  000002  037024          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
142126  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
142134  001003                          BNE     1$              ;;BRANCH IF NO
142136  105037  142156                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
142142  000406                          BR      $TYPEX          ;;EXIT
142144  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
142152  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
142154  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
142156  000000          $CHARCNT:.WORD  0                       ;;CHARACTER COUNT STORAGE
142160  000207          $TYPEX: RTS     PC
21815                           .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
                                ;;**************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
```

BINARY TO OCTAL (ASCII) AND TYPE

```
                                        ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                        ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                        ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                        ;*      .BYTE   M               ;;M=1 OR 0
                                        ;*                                      ;;1=TYPE LEADING ZEROS
                                        ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                        ;*
                                        ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                        ;*$TYPOS OR $TYPOC
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPON                   ;;CALL FOR TYPEOUT
                                        ;*
                                        ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                        ;*CALL:
                                        ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                        ;*      TYPOC                   ;;CALL FOR TYPEOUT
142162  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
142166  116637  000001  142405          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
142174  112637  142407                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
142200  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
142204  000406                          BR      $TYPON
142206  112737  000001  142405  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
142214  112737  000006  142407          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
142222  112737  000005  142404  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
142230  010346                          MOV     R3,-(SP)        ;;SAVE R3
142232  010446                          MOV     R4,-(SP)        ;;SAVE R4
142234  010546                          MOV     R5,-(SP)        ;;SAVE R5
142236  113704  142407                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
142242  005404                          NEG     R4
142244  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
142250  110437  142406                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
142254  113704  142405                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
142260  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
142264  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
142266  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
142270  000404                          BR      3$              ;;GO DO MSB
142272  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
142274  006105                          ROL     R5
142276  006105                          ROL     R5
142300  010503                          MOV     R5,R3
142302  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
142304  105337  142406                  DECB    $OMODE          ;;TYPE THIS DIGIT?
142310  100016                          BPL     7$              ;;BR IF NO
142312  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
142316  001002                          BNE     4$              ;;TEST FOR 0
142320  005704                          TST     R4              ;;SUPPRESS THIS 0?
142322  001403                          BEQ     5$              ;;BR IF YES
142324  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
142326  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
142332  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
142336  110337  142402                  MOVB    R3,8$           ;;SAVE FOR TYPING
142342  104401  142402                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
142346  105337  142404          7$:     DECB    $OCNT           ;;COUNT BY 1
```

BINARY TO OCTAL (ASCII) AND TYPE

```
142352  003347                          BGT     2$              ;;BR IF MORE TO DO
142354  002402                          BLT     6$              ;;BR IF DONE
142356  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
142360  000744                          BR      2$              ;;GO DO THE LAST DIGIT
142362  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
142364  012604                          MOV     (SP)+,R4        ;;RESTORE R4
142366  012603                          MOV     (SP)+,R3        ;;RESTORE R3
142370  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
142376  012616                          MOV     (SP)+,(SP)
142400  000002                          RTI                     ;;RETURN
142402     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
142403     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
142404     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
142405     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
142406  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
 21816                          .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
                                ;;**************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
                                ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                                ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                                ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                ;*REPLACED WITH SPACES.
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                ;*      TYPDS                   ;;GO TO THE ROUTINE
142410                          $TYPDS:
142410  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
142412  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
142414  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
142416  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
142420  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
142422  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
142426  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
142432  100004                          BPL     1$              ;;BR IF INPUT IS POS.
142434  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
142436  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
142444  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
142446  012703  142624                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
142452  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
142456  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
142460  016001  142614                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
142464  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
142466  002402                          BLT     4$              ;;BR IF DONE
142470  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
142472  000774                          BR      3$
142474  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
142476  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
142500  001002                          BNE     5$              ;;FALL THROUGH IF 0
142502  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
142504  100407                          BMI     7$              ;;BR IF YES
142506  106316                  5$:     ASLB    (SP)            ;;MSD?
142510  103003                          BCC     6$              ;;BR IF NO
142512  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
142520  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
142524  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
142530  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
142532  005720                          TST     (R0)+           ;;JUST INCREMENTING
```

CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```
142534  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
142540  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
142542  003002                          BGT     8$              ;;GO TO EXIT
142544  010502                          MOV     R5,R2           ;;GET THE LSD
142546  000764                          BR      6$              ;;GO CHANGE TO ASCII
142550  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
142552  100003                          BPL     9$              ;;BR IF NO
142554  116663  177777  177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
142562  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
142564  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
142566  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
142570  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
142572  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
142574  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
142576  104401  142624                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
142602  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
142610  012616                          MOV     (SP)+,(SP)
142612  000002                          RTI                     ;;RETURN TO USER
142614  023420                  $DTBL:  10000.
142616  001750                          1000.
142620  000144                          100.
142622  000012                          10.
142624                          $DBLK:  .BLKW   4
 21817                          .SBTTL  TTY INPUT ROUTINE
                                ;;******************************************************************
                                .ENABL  LSB
                                ;;******************************************************************
                                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                ;*WHEN OPERATING IN TTY FLAG MODE.
142634  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
142642  001074                          BNE     15$             ;;BRANCH IF NO
142644  105777  036274                  TSTB    @$TKS           ;;CHAR THERE?
142650  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
142652  117746  036270                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
142656  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
142662  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
142666  001062                          BNE     15$             ;;NO, RETURN TO USER
142670  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
142676  001456                          BEQ     15$             ;;BRANCH IF YES
142700  104401  143371                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
142704  104401  143376          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
142710  013746  000176                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
142714  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
142716  104401  143407                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
142722  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
142724  005046                          CLR     -(SP)           ;;THE NEW SWR
142726  105777  036212          7$:     TSTB    @$TKS           ;;CHAR THERE?
142732  100375                          BPL     7$              ;;IF NOT TRY AGAIN
142734  117746  036206                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
142740  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
142744  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
142750  001005                          BNE     10$             ;;BRANCH IF NOT
142752  104401  143364                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
142756  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
142762  000757                          BR      19$             ;;LET'S TRY IT AGAIN
```

TTY INPUT ROUTINE

```
142764  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
142770  001022                  BNE     16$             ;;BRANCH IF NO
142772  005766  000004          TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
142776  001403                  BEQ     11$             ;;BRANCH IF YES
143000  016677  000002  036132  MOV     2(SP),@SWR      ;;SAVE NEW SWR
143006  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
143012  104401  001175          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
143016  123727  001135  000001  CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
143024  001003                  BNE     15$             ;;BRANCH IF NOT
143026  012777  000100  036110  MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
143034  000002          15$:    RTI                     ;;RETURN
143036  004737  142040          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
143042  021627  000060          CMP     (SP),#60        ;;CHAR < 0?
143046  002420                  BLT     18$             ;;BRANCH IF YES
143050  021627  000067          CMP     (SP),#67        ;;CHAR > 7?
143054  003015                  BGT     18$             ;;BRANCH IF YES
143056  042726  000060          BIC     #60,(SP)+       ;;STRIP-OFF ASCII
143062  005766  000002          TST     2(SP)           ;;IS THIS THE FIRST CHAR
143066  001403                  BEQ     17$             ;;BRANCH IF YES
143070  006316                  ASL     (SP)            ;;NO, SHIFT PRESENT
143072  006316                  ASL     (SP)            ;;   CHAR OVER TO MAKE
143074  006316                  ASL     (SP)            ;;   ROOM FOR NEW ONE.
143076  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
143102  056616  177776          BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
143106  000707                  BR      7$              ;;GET THE NEXT ONE
143110  104401  001174          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
143114  000720                  BR      20$             ;;SIMULATE CONTROL-U
                        .DSABL  LSB
                        ;;*****************************************************************
                        ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                        ;*CALL:
                        ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                        ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                        ;*                              ;;WITH PARITY BIT STRIPPED OFF
                        ;
143116  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
143120  016666  000004  000002  MOV     4(SP),2(SP)     ;;SAVE THE PS
143126  105777  036012          1$:     TSTB    @$TKS           ;;WAIT FOR
143132  100375                  BPL     1$              ;;A CHARACTER
143134  117766  036006  000004  MOVB    @$TKB,4(SP)     ;;READ THE TTY
143142  042766  177600  000004  BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
143150  026627  000004  000023  CMP     4(SP),#23       ;;IS IT A CONTROL-S?
143156  001013                  BNE     3$              ;;BRANCH IF NO
143160  105777  035760          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
143164  100375                  BPL     2$              ;;LOOP UNTIL ITS THERE
143166  117746  035754          MOVB    @$TKB,-(SP)     ;;GET CHARACTER
143172  042716  177600          BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
143176  022627  000021          CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
143202  001366                  BNE     2$              ;;IF NOT DISCARD IT
143204  000750                  BR      1$              ;;YES, RESUME
143206  026627  000004  000021  3$:     CMP     4(SP),#$XON     ;;IS IT A RANDOM XON?           ;RAN001
143214  001744                  BEQ     1$              ;;BRANCH IF YES                 ;RAN001
143216  026627  000004  000140  CMP     4(SP),#140      ;;IS IT UPPER CASE?
143224  002407                  BLT     4$              ;;BRANCH IF YES
143226  026627  000004  000175  CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
143234  003003                  BGT     4$              ;;BRANCH IF YES
143236  042766  000040  000004  BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
```

TTY INPUT ROUTINE

```
143244  000002                  4$:     RTI                     ;;GO BACK TO USER
                                ;;**************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
                                ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
143246  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
143250  012703  143354          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
143254  022703  143364          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
143260  101405                          BLOS    4$              ;;BR IF YES
143262  104410                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
143264  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
143266  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
143272  001003                          BNE     3$              ;;SKIP IF NOT
143274  104401  001174          4$:     TYPE    ,$QUES          ;;TYPE A '?'
143300  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
143302  111337  143352          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
143306  104401  143352                  TYPE    ,9$
143312  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
143316  001356                          BNE     2$              ;;LOOP IF NOT RETURN
143320  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
143324  104401  001176                  TYPE    ,$LF            ;;TYPE A LINE FEED
143330  012603                          MOV     (SP)+,R3        ;;RESTORE R3
143332  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
143334  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
143342  012766  143354  000004          MOV     #$TTYIN,4(SP)
143350  000002                          RTI                     ;;RETURN
143352     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
143353     000                          .BYTE   0               ;;TERMINATOR
143354                          $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
143364     136     125     015  $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
143371     136     107     015  $CNTLG: .ASCIZ  /↑G/<15><12>    ;;CONTROL "G"
143376     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
143407     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
21818                           .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
                                ;;**************************************************************
                                ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                ;*CHANGE IT TO BINARY.
                                ;*CALL:
                                ;*      RDOCT                   ;;READ AN OCTAL NUMBER
                                ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
143420  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
143422  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
143430  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
143432  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
143434  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
143436  104411                  1$:     RDLIN                   ;;READ AN ASCIZ LINE
143440  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
143442  005001                          CLR     R1              ;;CLEAR DATA WORD
143444  005002                          CLR     R2
143446  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
143450  001412                          BEQ     3$              ;;IF ZERO GET OUT
143452  006301                          ASL     R1              ;;*2
143454  006102                          ROL     R2
143456  006301                          ASL     R1              ;;*4
```

READ AN OCTAL NUMBER FROM THE TTY

```
143460  006102                          ROL     R2
143462  006301                          ASL     R1              ;;*8
143464  006102                          ROL     R2
143466  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
143472  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
143474  000764                          BR      2$              ;;LOOP
143476  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
143500  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
143504  010237  143520                  MOV     R2,$HIOCT
143510  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
143512  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
143514  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
143516  000002                          RTI                     ;;RETURN
143520  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
21819                           .SBTTL  TRAP DECODER
                                ;;**************************************************************
                                ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
                                ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
                                ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
                                ;*GO TO THAT ROUTINE.
143522  010046                  $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
143524  016600  000002                  MOV     2(SP),R0        ;;GET TRAP ADDRESS
143530  005740                          TST     -(R0)           ;;BACKUP BY 2
143532  111000                          MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
143534  006300                          ASL     R0              ;;POSITION FOR INDEXING
143536  016000  143556                  MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
143542  000200                          RTS     R0              ;;GO TO ROUTINE
                                ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
143544  011646                  $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
143546  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
143554  000002                          RTI                     ;;RESTORE THE PSW
                                .SBTTL  TRAP TABLE
                                ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                                ;*BY THE "TRAP" INSTRUCTION.
                                ;       ROUTINE
                                ;       -------
143556  143544                  $TRPAD: .WORD   $TRAP2
143560  141626                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
143562  142206                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
143564  142162                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
143566  142222                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
143570  142410                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
143572  142704                          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
143574  142634                          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
143576  143116                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
143600  143246                          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
143602  143420                          $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
21820                           .SBTTL  POWER DOWN AND UP ROUTINES
                                ;;**************************************************************
                                ;POWER DOWN ROUTINE
143604  012737  143744  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
143612  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
143620  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
143622  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
143624  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
143626  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
143630  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
```

POWER DOWN AND UP ROUTINES

```
143632  010546                          MOV     R5,-(SP)          ;;PUSH R5 ON STACK
143634  017746  035300                  MOV     @SWR,-(SP)        ;;PUSH @SWR ON STACK
143640  010637  143750                  MOV     SP,$SAVR6         ;;SAVE SP
143644  012737  143656  000024          MOV     #$PWRUP,@#PWRVEC  ;;SET UP VECTOR
143652  000000                          HALT
143654  000776                          BR      .-2               ;;HANG UP
                        ;;*****************************************************************
                        ;POWER UP ROUTINE
143656  012737  143744  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC  ;;SET FOR FAST DOWN
143664  013706  143750                  MOV     $SAVR6,SP         ;;GET SP
143670  005037  143750                  CLR     $SAVR6            ;;WAIT LOOP FOR THE TTY
143674  005237  143750          1$:     INC     $SAVR6            ;;WAIT FOR THE INC
143700  001375                          BNE     1$                ;;OF  WORD
143702  012677  035232                  MOV     (SP)+,@SWR        ;;POP STACK INTO @SWR
143706  012605                          MOV     (SP)+,R5          ;;POP STACK INTO R5
143710  012604                          MOV     (SP)+,R4          ;;POP STACK INTO R4
143712  012603                          MOV     (SP)+,R3          ;;POP STACK INTO R3
143714  012602                          MOV     (SP)+,R2          ;;POP STACK INTO R2
143716  012601                          MOV     (SP)+,R1          ;;POP STACK INTO R1
143720  012600                          MOV     (SP)+,R0          ;;POP STACK INTO R0
143722  012737  143604  000024          MOV     #$PWRDN,@#PWRVEC  ;;SET UP THE POWER DOWN VECTOR
143730  012737  000340  000026          MOV     #340,@#PWRVEC+2   ;;PRIO:7
143736  104401                          TYPE                      ;REPORT THE POWER FAILURE
143740  143752          $PWRMG: .WORD   $POWER            ;;POWER FAIL MESSAGE POINTER
143742  000002                  RTI
143744  000000          $ILLUP: HALT                      ;;THE POWER UP SEQUENCE WAS STARTED
143746  000776                  BR      .-2               ;; BEFORE THE POWER DOWN WAS COMPLETE
143750  000000          $SAVR6: 0                         ;;PUT THE SP HERE
143752     015     012     120  $POWER: .ASCIZ  <15><12>"POWER"
                                .EVEN
21822           000001          .END
```

Symbol table

```
ABASE = 000000        AMADR3= 000000        BIT14  = 040000        DATBO   124574        D4      052760
ABOEXT  111314        AMADR4= 000000        BIT15  = 100000        DATI    125124        D5      052774
ABORT   141220        AMAMS1= 000000        BIT2   = 000004        DATVER  140100        D6      053004
ABORTC  141252        AMAMS2= 000000        BIT3   = 000010        DATVFR  140062        D7      053010
ABORTE  141316        AMAMS3= 000000        BIT4   = 000020        DAT1    140112        EEPAS   003034
ABORTI  107634        AMAMS4= 000000        BIT5   = 000040        DCOUNT  114144        EMTOA   030464
ABORTR  107676        AMSGAD= 000000        BIT6   = 000100        DDISP = 177570        EMTOB   030472
ABORTZ  141342        AMSGLG= 000000        BIT7   = 000200        DELAY   116754        EMTSAV  004024
ABORTO  044016        AMSGTY= 000000        BIT8   = 000400        DETFPA  140132        EMTVEC= 000030
ABORT7  044204        AMTYP1= 000000        BIT9   = 001000        DH1     134621        EM1     127002
ABROUT  111146        AMTYP2= 000000        BNO      043746        DH105   135562        EM10    127254
ACDW1 = 000000        AMTYP3= 000000        BN1      044136        DH115   135614        EM100   132570
ACDW2 = 000000        AMTYP4= 000000        BPTOA    030660        DH134   135671        EM101   132626
ACPUOP= 000000        APASS = 000000        BPTOB    030666        DH24    135105        EM102   132661
ACTCHS  002724        APRIOR= 000000        BPTVEC= 000014        DH27    135151        EM103   132743
AC0    =%000000       APTCSU= 000040        BTER     051362        DH4     134646        EM104   133016
AC1    =%000001       APTENV= 000001        BTEXP    003100        DH41    135214        EM105   133066
AC2    =%000002       APTSIZ= 000200        BTGO     051162        DH43    135265        EM106   133141
AC3    =%000003       APTSPO= 000100        BTRES    003110        DH47    135365        EM107   133217
AC4    =%000004       ASWREG= 000000        BTTST    051662        DH5     134733        EM11    127275
AC5    =%000005       ATESTN= 000000        BTTSTE   051710        DH65    135452        EM110   133253
AC6    =%000006       AUNIT = 000000        BYPARR   106114        DH7     135032        EM111   133315
AC7    =%000007       AUSWR = 000000        CCHPAS   003032        DH72    135516        EM112   133373
ADDLSB  114154        AVECT1= 000000        CCR    = 177746        DISPLA  001142        EM113   133456
ADDT    052264        AVECT2= 000000        CCRTBL   111106        DISPRE  000174        EM114   133542
ADDTRP  137720        A126    025246        CHECK    015342        DMAPAR  124040        EM115   133563
ADDW0 = 000000        BA      002670        CHECK1   015420        DMARD   137452        EM116   133633
ADDW1 = 000000        BACDAT  105606        CHECK2   015476        DMATRN  137370        EM117   133714
ADDW10= 000000        BA2     002710        CHECK7   053606        DPAREN  124100        EM12    127317
ADDW11= 000000        BCR    = 177524        CHEC10   054032        DSWR   = 177570        EM120   133757
ADDW12= 000000        BCSR   = 177520        CHEC26   057214        DT1     136026        EM121   134025
ADDW13= 000000        BDR    = 177524        CHEC27   057556        DT105   136304        EM122   134123
ADDW14= 000000        BFA      052316        CHEC30   060122        DT115   136314        EM123   134151
ADDW15= 000000        BFAC1    052060        CHEC32   060554        DT130   136330        EM124   134203
ADDW2 = 000000        BFAC2    052104        CHEC33   061050        DT134   136336        EM125   134275
ADDW3 = 000000        BFAC3    052130        CHEK7    053612        DT14    136074        EM126   134334
ADDW4 = 000000        BFAC4    052154        CHK10    054050        DT17    136106        EM127   134371
ADDW5 = 000000        BFAC5    052202        CHK7     053620        DT24    136120        EM13    127345
ADDW6 = 000000        BFAE     052356        CH10     054036        DT27    136130        EM130   134433
ADDW7 = 000000        BFB      052354        CKSWR  = 104407        DT35    136142        EM131   134460
ADDW8 = 000000        BGNTLP   113472        CMPRTN   066176        DT4     136034        EM132   134510
ADDW9 = 000000        BIT0   = 000001        COUNT    003120        DT41    136154        EM133   134546
ADEVCT= 000000        BIT00  = 000001        CPEREG=  177766        DT43    136164        EM134   134567
ADEVM = 000000        BIT01  = 000002        CR     = 000015        DT47    136200        EM14    127374
AENV  = 000000        BIT02  = 000004        CRLF   = 000200        DT5     136046        EM15    127417
AENVM = 000000        BIT03  = 000010        CSR1     002664        DT50    136212        EM16    127453
AFATAL= 000000        BIT04  = 000020        CSR12    002704        DT51    136224        EM17    127520
ALLCTR  003152        BIT05  = 000040        CSR2     002666        DT52    136236        EM2     127036
ALLEND  125304        BIT06  = 000100        CSR22    002706        DT64    136250        EM20    127564
ALROTS  021532        BIT07  = 000200        CURADD   114170        DT65    136262        EM21    127617
ALR1TS  021610        BIT08  = 000400        CURDAT   114162        DT7     136062        EM22    127662
ALR2TS  021666        BIT09  = 001000        C121A    023262        DT75    136272        EM23    127721
ALR3TS  021744        BIT1   = 000002        C121B    023302        DVDSUB  070236        EM24    127775
ALR4TS  022022        BIT10  = 002000        C121C    023322        DVFSUB  067244        EM25    130040
ALR5TS  022100        BIT11  = 004000        DAPABO   107350        D1      052726        EM26    130100
AMADR1= 000000        BIT12  = 010000        DATA     002674        D2      052740        EM27    130147
AMADR2= 000000        BIT13  = 020000        DATA2    002714        D3      052742        EM3     127050
```

K1

Symbol table

```
EM30    130207          EXBAD2  115266          GPR4TS  005374          KIPAR4= 172350          MBTCC   015614
EM31    130261          EXITST  114134          GPR5TS  005452          KIPAR5= 172352          MBTD    032760
EM32    130306          EXPBDT  111126          GPR6TS  005530          KIPAR6= 172354          MBTE    051712
EM33    130347          EXPDAT  114146          GTSWR = 104406          KIPAR7= 172356          MBTF    032770
EM34    130411          EXPIR1  034344          HITMIS= 177752          KIPDR0= 172300          MBTO    033002
EM35    130451          EXPTBL  105634          HMPARR  105404          KIPDR1= 172302          MBTOA   033032
EM36    130477          EXPWDT  111116          HOP10   067506          KIPDR2= 172304          MBTOB   033034
EM37    130543          EXTFPA  140172          HOP11   070500          KIPDR3= 172306          MBTOC   033046
EM4     127062          FINNOP  021252          HOP12   071536          KIPDR4= 172310          MBTOD   033072
EM40    130605          FIN1    052444          HOP13   072564          KIPDR5= 172312          MBTOE   033074
EM41    130652          FIN10   054114          HOP14   073660          KIPDR6= 172314          MBTOF   033106
EM42    130736          FIN11   054220          HOP15   075354          KIPDR7= 172316          MBT1    051140
EM43    130764          FIN116  022742          HOP16   075670          KMCR  = 177734          MBT2    051142
EM44    131015          FIN117  023056          HOP17   076352          LASTCH  116110          MBT2A   051154
EM45    131055          FIN120  023200          HOP18   100612          LCDSUB  101102          MBT8    051370
EM46    131137          FIN121  023574          HOP19   101230          LCFSUB  100464          MBT8A   051424
EM47    131227          FIN122  024044          HOP20   102650          LDPARS  136736          MBT8B   051452
EM5     127122          FIN125  025206          HOP21   103734          LDPDRS  136766          MBT8C   051500
EM50    131314          FIN126  026246          HOP22   104376          LEDS    123016          MBT8D   051526
EM51    131337          FIN127  026706          HOP44   066324          LF    = 000012          MBT8E   051556
EM52    131371          FIN13   054434          HT    = 000011          LKS   = 177546          MBT8F   051606
EM53    131427          FIN130  027576          ILAOA   030760          LKSFL   002722          MBT8FG  051622
EM54    131463          FIN14   054604          ILBOB   030766          LKSINT  137062          MBT8I   051646
EM55    131521          FIN15   055022          ILL     053230          LOOP    004740          MB66    011770
EM56    131555          FIN16   055212          ILLBOA  031054          LOOPIN  003154          MCB44   011634
EM57    131601          FIN17   055424          ILLBOB  031062          LOST    052424          MCLRD   076416
EM6     127155          FIN2    052544          ILLOP1  053030          LOWADD  003016          MCLRI   076504
EM60    131633          FIN20   055622          ILLOP2  053124          LSTADD  114166          MCMPD   066014
EM61    131701          FIN21   056042          INITMM  136574          LXPSUB  102400          MCTSCC  016066
EM62    131730          FIN22   056226          INQ22   125446          MACCC   016216          MDAO    010406
EM63    132001          FIN23   056362          INTERR  110106          MACE    051322          MDCCC   016004
EM64    132032          FIN24   056562          INTRPC  110052          MACO    051212          MDDSUB  075054
EM65    132071          FIN26   057316          IOTOA   030366          MACOA   051216          MDFSUB  073350
EM66    132141          FIN27   057660          IOTOB   030374          MAC1    051226          MDIVD   067506
EM67    132200          FIN30   060224          IOTVEC= 000020          MAC2    051242          MDIVF   066324
EM7     127220          FIN31   060364          IOXXX   033520          MAC3    051256          MDMO    010362
EM70    132235          FIN32   060656          J11FLT  140257          MAC4    051272          MDM27   010472
EM71    132266          FIN33   061152          KDPAR0= 172360          MAC5    051306          MDSO    010436
EM72    132322          FIN4    053134          KDPAR1= 172362          MADCC   016142          MEMK    123434
EM73    132360          FIN5    053240          KDPAR2= 172364          MAIREG= 177750          MEMQ    123464
EM74    132404          FIN6    053440          KDPAR3= 172366          MALCC   016770          MEMTO   030420
EM75    132435          FIN7    053664          KDPAR4= 172370          MARCC   017050          MET     032442
EM76    132466          FLAG    003030          KDPAR5= 172372          MASK    003160          META    032500
EM77    132536          FLO     003062          KDPAR6= 172374          MA11    011010          METB    032510
ENDHRT  105576          FLOAT   003052          KDPAR7= 172376          MA55    011714          METD    032520
ENDLUP  114110          FMPARR  105214          KDPDR0= 172320          MBB11   011106          METF    032530
ENDMOV  114172          FPAFLT  140174          KDPDR1= 172322          MBB22   011370          METO    032550
ENDTAG  115314          FPAOPT  140160          KDPDR2= 172324          MBCCC   015664          METOA   032614
ENDTLP  114060          FPVEC = 000244          KDPDR3= 172326          MBCOO   010612          METOB   032624
ERR     140060          FRSTST  004742          KDPDR4= 172330          MBC11   011166          METOC   032636
ERRCNT  003022          FSTADD  114164          KDPDR5= 172332          MBC22   011456          METOD   032662
ERRFP   140056          FWDSEQ  114152          KDPDR6= 172334          MBIOO   010532          METOE   032672
ERROR = 104000          GOODAD  003020          KDPDR7= 172336          MBPTO   030614          METOF   032704
ERROUT  104536          GPROTS  005104          KIPAR0= 172340          MBSCC   015736          MFA     051714
ERRVEC= 000004          GPR1TS  005162          KIPAR1= 172342          MBT     032724          MFACU   051712
ERTYPE  136356          GPR2TS  005240          KIPAR2= 172344          MBTA    032746          MFSRCM  061154
EXBAD   114124          GPR3TS  005316          KIPAR3= 172346          MBTB    032750          MIALL   033320
```

Symbol table

MIALLA	033344	MJR77A	014776	MLDC2	077644	MRTB	031550	MST6	010240
MIALLB	033346	MJSI	033400	MLDDM2	061240	MRTE	031560	MST7	010306
MIALLD	033356	MJSIA	033432	MLDDM3	061342	MRTF	031570	MSW37	012614
MIALLF	033366	MJSIB	033434	MLDDM4	061476	MRTO	031602	MSXP	104076
MIL	033120	MJSIC	033446	MLDDM5	061616	MRTOA	031632	MSXT	017302
MILA	033144	MJSID	033472	MLDDM6	061722	MRTOB	031634	MSXTCC	017140
MILAO	030712	MJSIE	033474	MLDDM7	062016	MRTOC	031646	MS11	011046
MILB	033146	MJSIF	033506	MLDM27	062120	MRTOD	031672	MS22	011320
MILD	033156	MJSR	013716	MLDSUB	072322	MRTOE	031674	MS33	011550
MILF	033166	MJSRA	014034	MLFSUB	071274	MRTOF	031706	MS77	012040
MILLBO	031012	MJSRB	014200	MLS1	076550	MRTS	015002	MTP	031720
MILLO	030224	MJSRC	014346	MLS2	076662	MSB	064216	MTPA	032122
MILLOA	030270	MJSR1	014036	MLS3	076774	MSBCC	016416	MTPAA	032176
MILLOB	030276	MJSR1A	014050	MLS4	077124	MSBCCC	016466	MTPAE	032226
MILO	033200	MJSR1B	014074	MLS5	077240	MSCD	102650	MTPAH	032174
MILOA	033232	MJSR2	013754	MLS6	077370	MSCF	103734	MTPAL	032164
MILOB	033234	MJSR2A	013766	MLS7	077510	MSDF	075670	MTPB	031752
MILOC	033246	MJSR2B	014012	MLXP	101230	MSER =	177744	MTPF	031772
MILOD	033272	MJSR3	014202	MMARK	017554	MSFD	075354	MTPO	032004
MILOE	033274	MJSR3A	014214	MMODD	073660	MSFDI	076352	MTPOA	032034
MILOF	033306	MJSR3B	014240	MMODF	072564	MSOB	017500	MTPOB	032036
MIOT	032246	MJSR4	014116	MMRL5	012452	MSPAA	007640	MTPOC	032050
MIOTA	032270	MJSR4A	014130	MMR0 =	177572	MSPAU	027576	MTPOD	032074
MIOTB	032272	MJSR4B	014154	MMR1 =	177574	MSPB	005732	MTPOE	032076
MIOTD	032302	MJSR5	014350	MMR2 =	177576	MSPBB	007724	MTPOF	032110
MIOTF	032312	MJSR5A	014362	MMR3 =	172516	MSPC	005756	MTPQ	031762
MIOTO	030322	MJSR5B	014406	MMU	137552	MSPD	006006	MTPR	031750
MITO	032324	MJSR6	014264	MMULD	071536	MSPEO	006044	MTRPO	030516
MITOA	032354	MJSR6A	014276	MMULF	070500	MSPF	006106	MTRY	027740
MITOB	032356	MJSR6B	014322	MMUTRP	137724	MSPG	006150	MTRYA	030014
MITOC	032370	MJSR7	014432	MMVCC	015550	MSPH	006206	MTRYB	030034
MITOD	032414	MJSR7A	014444	MMVEC =	000250	MSPI	006252	MTRYM	030066
MITOE	032416	MJSR7E	014470	MM11	010700	MSPJ	006312	MTSO	015550
MITOF	032430	MJU1	012772	MM22	011246	MSPL	006432	MTT	031176
MJ	012720	MJU1A	013004	MNCCCC	016300	MSPM	006500	MTTA	031232
MJP	013556	MJU2	012734	MNGOP	063406	MSPN	006556	MTTB	031234
MJP17	013566	MJU2A	012746	MNNRM1	062226	MSPO	006624	MTTD	031244
MJP27	013610	MJU2B	012756	MNNRM2	062506	MSPP	006730	MTTE	031254
MJP27A	013622	MJU3	013016	MNNRM3	062660	MSPQ	007026	MTTR	031360
MJP37	013674	MJU3A	013030	MNNRM4	063022	MSPR	007134	MTTRA	031424
MJP37A	013706	MJU3B	013040	MNRM	064360	MSPS	007222	MTTRB	031426
MJP67	013632	MJU4	013112	MODE1	046034	MSPT	007270	MTTRC	031440
MJP67A	013644	MJU4A	013124	MODE2	047414	MSPU	007306	MTTRD	031476
MJP67B	013660	MJU4B	013136	MODGAR	073650	MSPV	007406	MTTRE	031500
MJP77	013662	MJU5	013052	MRLB1	012200	MSPVO	007352	MTTRF	031512
MJP77E	013716	MJU5A	013064	MRLCC	016552	MSPX	007452	MTTS	031266
MJRA	014474	MJU5B	013076	MRLO	012126	MSPY	007522	MTTSA	031322
MJR27	014530	MJU6	013152	MRL2	012260	MSPZ	007570	MTTSB	031326
MJR27A	014564	MJU6A	013164	MRL3	012336	MSPO	005710	MTTSD	031336
MJR27B	014604	MJU7	013204	MRL4	012412	MSTB3	007774	MTTSE	031346
MJR37	014666	MJU7E	013216	MRL6	012514	MSTO	031102	MTTSQ	031324
MJR37A	014722	MJ2	012770	MRL7	012552	MSTOE	031156	MUVAD	064500
MJR6A	014662	MJ5	013150	MRRB1	012672	MSTOEE	031164	MXDF1	063142
MJR6B	014664	MJ7	013200	MRRCC	016632	MST4	010046	MXOR	017420
MJR67	014606	MLCD	100612	MRRO	012650	MST4B	010104	MXRCC	017210
MJR67A	014642	MLCF	077760	MRT	031524	MST5	010144	M2	004766
MJR77	014742	MLDC	065376	MRTA	031546	MST5B	010176	M3	005002

Symbol table

```
M4      005022          POLY   = 120001        SDPAR5= 172272        SW01   = 000002        TAB42   003724
M5      005042          PROCNT  023126        SDPAR6= 172274        SW02   = 000004        TAB43   003734
M6      005062          PR0    = 000000        SDPAR7= 172276        SW03   = 000010        TAB45   003744
NEWADD  003026          PR1    = 000040        SDPDR0= 172220        SW04   = 000020        TAB46   003754
NEWDAT  114160          PR2    = 000100        SDPDR1= 172222        SW05   = 000040        TAB47   003764
NOABRT  141356          PR3    = 000140        SDPDR2= 172224        SW06   = 000100        TAB47A  003774
NOFPA   140166          PR4    = 000200        SDPDR3= 172226        SW07   = 000200        TAB48   004004
NOTOK   140002          PR5    = 000240        SDPDR4= 172230        SW08   = 000400        TAB49   004014
NULL   = 000000         PR6    = 000300        SDPDR5= 172232        SW09   = 001000        TAB5    003274
NXMFIN  047636          PR7    = 000340        SDPDR6= 172234        SW1    = 000002        TAB5A   003304
NXMPAR  111350          PS     = 177776        SDPDR7= 172236        SW10   = 002000        TAB6    003314
NXMTRP  047244          PSW    = 177776        SEQ     003072        SW11   = 004000        TAB6A   003324
ODDXX   033620          PSWBTS  005612        SFDSUB  075550        SW12   = 010000        TAB7    003334
OK      137760          PWRVEC= 000024        SIMGOA  002702        SW13   = 020000        TAB8    003344
OKAY7   043056          Q22EN   003002        SIPAR0= 172240        SW14   = 040000        TAB9    003354
OKAY7A  043066          Q22INT  137350        SIPAR1= 172242        SW15   = 100000        TAPABO  107514
OKA7    043044          Q22SIZ  137070        SIPAR2= 172244        SW2    = 000004        TA114   021170
OK1     140006          RAMPAR  137012        SIPAR3= 172246        SW3    = 000010        TA116   022644
OK7     043026          RBUF   = 177562        SIPAR4= 172250        SW4    = 000020        TBITVE= 000014
ONOQ22  137314          RCSR   = 177560        SIPAR5= 172252        SW5    = 000040        TB114   021200
PAR     137674          RDCHR  = 104410        SIPAR6= 172254        SW6    = 000100        TC114   021210
PARAD1  045664          RDLIN  = 104411        SIPAR7= 172256        SW7    = 000200        TD114   021220
PARAD2  047276          RDOCT  = 104412        SIPDR0= 172200        SW8    = 000400        TEMP    002740
PARVA1  045716          RECDAT  114150        SIPDR1= 172202        SW9    = 001000        TE102   017760
PARVA2  047330          RECDST  003142        SIPDR2= 172204        SXPSUB  104256         TE103   020002
PARVA3  047446          RECFEC  003122        SIPDR3= 172206        TAB1    003234         TE104   020024
PAR1    137676          RECST   003132        SIPDR4= 172210        TAB10   003364         TE105   020046
PCR    = 177522         RESVEC= 000010        SIPDR5= 172212        TAB11   003374         TE106   020070
PDR     137652          RET1    015266        SIPDR6= 172214        TAB11A  003404         TE107   020112
PDR1    137654          RET2    015354        SIPDR7= 172216        TAB12   003414         TE110   020134
PHY1    046002          RET3    015432        SIXBIT  115460        TAB13   003424         TE111   020156
PIR    = 177772         RITEDA  1..156        SLEND   122620        TAB13B  003434         TE112   020200
PIRQ   = 177772         RTSE    015054        SLOC00  003012        TAB14   003444         TE113   020422
PIRQNX  034222          RTS1    015022        SLOC01  003014        TAB15   003454         TE113A  020642
PIRQT   125736          RTS6    015032        SPAU1   027614        TAB16   003464         TE114   021014
PIRQVE= 000240          RXXX    034040        SPAU2   027636        TAB17   003474         TE115   022164
PIRRTN  034522          R6     =%000006       SPAU3   027656        TAB18   003504         TE115A  022374
PIRTBL  034324          R7     =%000007       SPAU4   027702        TAB2    003244         TE115B  022404
PIRTEX  034224          SAVBR   002730        SPAU5   027716        TAB21   003514         TE115C  022412
PIRXXX  034166          SAVMR0  003042        SPAU6   027734        TAB22   003524         TE115D  022420
PI?1    034224          SAVMR1  003044        SPS     003074        TAB23   003534         TE115F  022426
P  2    034344          SAVMR2  003046        SPSJ    003076        TAB24   003544         TE116   022430
PLR2EX  034452          SAVPCR  002726        SR0    = 177572       TAB25   003554         TE116A  022704
PIR3    034452          SAVPOS  003156        SR1    = 177574       TAB26   003564         TE116B  022710
PIR3EX  034546          SAVSUP  003036        SR2    = 177576       TAB27   003574         TE116C  022722
PIR4    034546          SAVSWR  003050        SR3    = 172516       TAB28   003604         TE116D  022732
PIR5    034624          SAVUSE  003040        STACK  = 001100       TAB29   003614         TE117   022742
PIR5EX  035034          SAV30   004112        START   004024        TAB29A  003624         TE117A  023054
PIR6    035034          SAV32   004114        STBOT  = 001000       TAB3    003254         TE120   023060
PIR6EX  035136          SCDSUB  103470        STKLMT= 177774        TAB30   003634         TE120A  023134
PITBL1  034430          SCOPE  = 000004        STMOVI  113326        TAB31   003644         TE121   023210
PITBL2  035020          SDFSUB  076120        STMOVT  114450        TAB32   003654         TE122   023574
PI1     034474          SDPAR0= 172260        SUBT    052232        TAB33   003664         TE123   024046
PI2     034506          SDPAR1= 172262        SWR     001140        TAB34   003674         TE124   024266
PI3     034510          SDPAR2= 172264        SWREG   000176        TAB4    003264         TE125   024520
PLF0    043674          SDPAR3= 172266        SW0    = 000001       TAB40   003704         TE125A  024760
PLF1    044066          SDPAR4= 172270        SW00   = 000001       TAB41   003714         TE126   025206
```

Symbol table

TE126A	025254	TSFP5	053134	TST13	107044	TS14	047134	T123D	024254
TE126B	025730	TSFP6	053240	TST14	107360	TS15	050020	T123E	024256
TE127	026246	TSFP7	053440	TST15	107524	TS16	051024	T123F	024262
TE127A	026446	TSF10	054054	TST16	107752	TS16A	051016	T124A	024462
TE130	026710	TSF13	054424	TST17	110156	TS1822	046066	T124B	024472
TE130A	027166	TSF14	054574	TST2	104376	TS26D0	057236	T124C	024500
TF114	021230	TSF15	055012	TST20	110402	TS26D1	057246	T124D	024506
TG114	021240	TSF16	055202	TST21	111150	TS26D2	057256	T124E	024510
THRBIT	115726	TSF17	055414	TST22	111360	TS26D3	057266	T124F	024514
TH114	021250	TSF2	052514	TST23	111600	TS26D4	057276	T13FIN	046066
TIMDEL	120602	TSF20	055612	TST24	112136	TS26D5	057306	T14	047160
TIMEOU	053114	TSF21	056032	TST25	112714	TS27D0	057600	T14FIN	047532
TIMOUT	003000	TSF22	056216	TST26	113056	TS27D1	057610	T15	050070
TKVEC =	000060	TSF23	056352	TST27	114172	TS27D2	057620	T15A	050146
TMM16A	050534	TSF24	056552	TST3	104542	TS27D3	057630	T15FIN	050174
TMM16B	050414	TSF31	060354	TST30	115314	TS27D4	057640	UDPAR0=	177660
TMM16C	050444	TSF6	053430	TST31	115466	TS27D5	057650	UDPAR1=	177662
TMM16D	050474	TSF7	053624	TST32	115734	TS30D0	060144	UDPAR2=	177664
TMM16E	050524	TSLOOP	114556	TST33	116254	TS30D1	060154	UDPAR3=	177666
TMM16F	050526	TSMA	043116	TST34	116406	TS30D2	060164	UDPAR4=	177670
TM16A	051120	TSMB	043136	TST35	116772	TS30D3	060174	UDPAR5=	177672
TOUT	137542	TSMC	043146	TST36	117156	TS30D4	060204	UDPAR6=	177674
TPVEC =	000064	TSMMU0	015056	TST37	117464	TS30D5	060214	UDPAR7=	177676
TRAPVE=	000034	TSMMU1	035152	TST4	104642	TS32D0	060576	UDPDR0=	177620
TRPFLG	140054	TSMMU2	035236	TST40	117632	TS32D1	060606	UDPDR1=	177622
TRPOA	030562	TSMMU3	036436	TST41	120134	TS32D3	060626	UDPDR2=	177624
TRPOB	030570	TSMMU4	036736	TST42	120302	TS32D4	060636	UDPDR3=	177626
TRTVEC=	000014	TSMMU5	037666	TST43	120612	TS32D5	060646	UDPDR4=	177630
TRYMA	030126	TSMMU6	040020	TST44	120652	TS33D0	061072	UDPDR5=	177632
TRYMB	030160	TSMMU7	042524	TST45	120750	TS33D1	061102	UDPDR6=	177634
TRYMC	030206	TSMMU8	043152	TST46	121032	TS33D2	061112	UDPDR7=	177636
TSD31	060252	TSMMU9	043354	TST47	121246	TS33D3	061122	UFDFLG	004116
TSEND	115276	TSMM10	044254	TST5	105226	TS33D4	061132	UFDSET=	000001
TSFP1	052356	TSMM11	044636	TST50	121300	TS33D5	061142	UIPAR0=	177640
TSFP10	053664	TSMM12	045074	TST51	121644	TS6DA	053410	UIPAR1=	177642
TSFP11	054114	TSMM13	045462	TST52	122112	TS6DAT	053420	UIPAR2=	177644
TSFP12	054220	TSMM14	046530	TST53	122402	TS7	042774	UIPAR3=	177646
TSFP13	054270	TSMM15	047654	TST54	122620	TS7DA1	053634	UIPAR4=	177650
TSFP14	054434	TSMM16	050174	TST55	123142	TS7DA2	053644	UIPAR5=	177652
TSFP15	054606	TSMM6A	040126	TST56	123516	TS7DA4	053654	UIPAR6=	177654
TSFP16	055024	TSMM6B	040564	TST57	123764	TS7FIN	043150	UIPAR7=	177656
TSFP17	055214	TSMM6C	041324	TST6	105412	TS9FIN	044252	UIPDR0=	177600
TSFP2	052446	TSMM6D	042024	TST60	124136	TYPDS =	104405	UIPDR1=	177602
TSFP20	055426	TSM16A	050270	TST61	124336	TYPE =	104401	UIPDR2=	177604
TSFP21	055624	TSM16B	050316	TST62	125304	TYPOC =	104402	UIPDR3=	177606
TSFP22	056044	TSM16C	050342	TST63	125562	TYPON =	104404	UIPDR4=	177610
TSFP23	056230	TSM16D	050406	TST64	125746	TYPOS =	104403	UIPDR5=	177612
TSFP24	056364	TSM7	043104	TST65	126046	T10FIN	044636	UIPDR6=	177614
TSFP25	056564	TSM9	043510	TST66	126230	T11FIN	045074	UIPDR7=	177616
TSFP26	056722	TSTADD	003024	TST7	105652	T114	021146	UNXPIR	034450
TSFP27	057320	TSTEND	116770	TS10	044554	T116	022612	UQUIET	004120
TSFP3	052544	TSTLOC	003162	TS10D1	054064	T12FIN	045460	VIREOP	126756
TSFP30	057662	TSTLUP	113410	TS10D2	054074	T122A	023672	VIR1	045750
TSFP31	060226	TST1	004740	TS10D4	054104	T122B	024024	VIR2	047362
TSFP32	060366	TST10	106126	TS11	044754	T123A	024230	VIR3	047500
TSFP33	060660	TST11	106370	TS11D1	054210	T123B	024240	VMKOR	004122
TSFP4	052660	TST12	106644	TS12	045412	T123C	024246	VQBE1	002676

Symbol table

```
VQBE2   002716          $DDW10  001310          $ETEND  001324          $MSGTY  001200          $TIMES  001164
VQPR1   002700          $DDW11  001312          $FATAL  001202          $MSWR   143376          $TKB    001146
VQPR2   002720          $DDW12  001314          $FFLG   141624          $MTYP1  001231          $TKS    001144
WC      002672          $DDW13  001316          $FILLC  001156          $MTYP2  001235          $TMP0   001160
WC2     002712          $DDW14  001320          $FILLS  001155          $MTYP3  001241          $TMP1   001162
WLDTRP  140044          $DDW15  001322          $GDADR  001120          $MTYP4  001245          $TN   = 000067
XBUF  = 177566          $DDW2   001270          $GDDAT  001124          $MXCNT  140774          $TPB    001152
XCBIT   016710          $DDW3   001272          $GET42  140430          $NULL   001154          $TPFLG  001157
XCSR  = 177564          $DDW4   001274          $GTSWR  142704          $NWTST= 000000          $TPS    001150
XME100  017714          $DDW5   001276          $HD   = 000001          $OCNT   142404          $TRAP   143522
XME101  017736          $DDW6   001300          $HIBTS  000232          $OMODE  142406          $TRAP2  143544
$APTHD  000232          $DDW7   001302          $HIOCT  143520          $OVER   140752          $TRP  = 000013
$ATYC   141404          $DDW8   001304          $ICNT   001104          $PASS   001206          $TRPAD  143556
$ATY1   141360          $DDW9   001306          $ILLUP  143744          $PASTM  000240          $TSTM   000236
$ATY3   141366          $DEVCT  001210          $INTAG  001135          $POWER  143752          $TSTNM  001102
$ATY4   141376          $DEVM   001256          $ITEMB  001114          $PWRDN  143604          $TTYIN  143354
$AUTOB  001134          $DOAGN  140450          $LF     001176          $PWRMG  143740          $TYPDS  142410
$BASE   001254          $DTBL   142614          $LFLG   141623          $PWRUP  143656          $TYPE   141626
$BDADR  001122          $ENDAD  140440          $LPADR  001106          $QUES   001174          $TYPEC  142040
$BDDAT  001126          $ENDCT  140406          $LPERR  001110          $RDCHR  143116          $TYPEX  142160
$BELL   001170          $ENDMG  140457          $MADR1  001232          $RDLIN  143246          $TYPOC  142206
$CDW1   001260          $ENULL  140454          $MADR2  001236          $RDOCT  143420          $TYPON  142222
$CDW2   001262          $ENV    001220          $MADR3  001242          $RDSZ = 000010          $TYPOS  142162
$CHARC  142156          $ENVM   001221          $MADR4  001246          $RTNAD  140452          $UNIT   001212
$CKSWR  142634          $EOP    140340          $MAIL   001200          $SAVR6  143750          $UNITM  000242
$CMTAG  001100          $EOPCT  140400          $MAMS1  001230          $SCOPE  140474          $USWR   001224
$CM3  = 000000          $ERFLG  001103          $MAMS2  001234          $SETUP= 000137          $VECT1  001250
$CM4  = 000002          $ERMAX  001115          $MAMS3  001240          $STUP = 177777          $VECT2  001252
$CNTLG  143371          $ERROR  140776          $MAMS4  001244          $SVLAD  140716          $XOFF = 000023
$CNTLU  143364          $ERRPC  001116          $MBADR  000234          $SVPC = 000232          $XON  = 000021
$CPUOP  001226          $ERRTB  001324          $MFLG   141622          $SWR  = 167400          $XTSTR  140514
$CRLF   001175          $ERTTL  001112          $MNEW   143407          $SWREG  001222          $$GET4= 000000
$DBLK   142624          $ESCAP  001166          $MSGAD  001214          $SWRMK= 000300          $OFILL  142405
$DDW0   001264          $ETABL  001220          $MSGLG  001216          $TESTN  001204          .$X   = 000232
$DDW1   001266

. ABS.  143762    000   (RW,I,GBL,ABS,OVR)
        000000    001   (RW,I,LCL,REL,CON)
Errors detected:  10

*** Assembler statistics


Work  file  reads: 471
Work  file writes: 403
Size of work file: 63968 Words  ( 250 Pages)
Size of core pool: 19714 Words  ( 75 Pages)
Operating  system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:09:05.30
COKDADO,COKDADO/NL:TOC/-SP=ORION.MLB/ML,COKDADO.MAC/DS:GBL
```